Owen Bishop

Guida al Commodore VIC 20





tecniche nuove

O. Bishop

Guida al Commodore VIC 20

Ringraziamenti

Desideriamo ringraziare la Commodore Business Machine, Slough, per i consigli e l'assistenza forniti durante la preparazione di questo libro.

Owen Bishop

Guida al Commodore VIC 20

tecniche nuove

EDIZIONE ORIGINALE

Get More from the VIC 20.

EDIZIONE ITALIANA

Traduzione dall'inglese di *Sandro Ratti.*

tel. (02) 6590351, telex 334647 TECHS I
ISBN 88 7081 167 0

Fotocomposizione: Grafica Quadrifoglio srl, Milano
Stampa: Polver, Milano

Indice

Introduzione

La scelta degli argomenti trattata in questo libro è stata fatta in base alle esigenze di tre categorie di lettori. Prima di tutto per coloro che iniziano a programmare, che hanno comprato un VIC 20 o che stanno per farlo. Il manuale fornito assieme alla macchina è utile per i primi passi, mentre questo libro vi accompagnerà in seguito facendovi anche divertire.

Poi vi sono i lettori che sono giunti all'uso del VIC 20 provenienti da un altro computer. Costoro già conoscono, forse anche abbastanza bene, il BASIC e desiderano applicare ciò che sanno al VIC. Questo libro vi renderà familiare il BASIC del VIC e vi mostrerà come usarlo effettivamente.

E infine vi sono i lettori attratti dalle prestazioni nella grafica a colori e dalla versatilità nella generazione di suoni. Questo libro vi aiuterà ad utilizzare queste opzioni in un modo divertente.

I programmi che abbiamo scelto sono stati scritti e verificati su di un VIC 20 non espanso. Se avete più di 3 kilobytes di memoria ausiliaria, è necessario in alcuni programmi cambiare alcuni valori, come è spiegato nell'appendice B.

Capitolo 1
VIC - Il computer come amico

Ci sono molte cose che si possono fare con un computer. Eccone alcune:

Giochi
Possono essere per uno o due giocatori o per tutta la famiglia. In molti casi in cui c'è un solo giocatore, questi sfida il computer, migliorando le sue prestazioni fino alla vittoria.

Musica
I programmi che riguardano la musica possono essere molto divertenti. Potete ascoltare il computer suonare o potete suonare voi stessi melodie di vostro gradimento usando la sua tastiera. Il VIC è particolarmente adatto per questo tipo di prestazioni.

Affari
Ci sono programmi per aiutarvi nella gestione degli affari domestici. Alcuni tengono la contabilità e vi aiutano nella pianificazione dei prossimi investimenti.

Figure
Tracciare disegni a colori sul video è un passatempo creativo. Sono disponibili molti programmi per iniziarvi all'"arte computerizzata'. Il VIC è ideale per realizzare figure colorate.

Archivi
Informazioni di ogni genere possono essere memorizzate nel computer, mischiate, analizzate, archiviate, pronte per essere disponibili ogniqualvolta lo vogliate. Questi programmi possono essere usati per memorizzare gli indirizzi, una lista della vostra collezione di francobolli, di spettacoli visti alla televisione, delle vostre ricette preferite o di informazioni per i clienti.

Educazione
Vi sono dozzine di modi in cui il computer può aiutare vostro figlio nella ricerca di nuove nozioni da acquisire.

Word processing
Se avete una stampante collegata al computer, potete usarlo come un word processor. Ciò rende molto semplice scrivere testi. La macchina è in grado di gestire parole che nessuna macchina da scrivere riesce a produrre. Per la casa o l'ufficio ottenete prestazioni a livelllo professionale con un minimo di sforzo e abilità.

Programmazione
E da ultimo ciò che è per molti l'aspetto più importante dell'home computer: fare eseguire alla macchina esattamente ciò che volete che faccia. Il VIC è più semplice da programmare di altri computer e questo libro vi spiega come fare.

Hardware e software

Quando togliete il computer dall'imballaggio trovate numerosi componenti. Il più voluminoso e costoso è il computer stesso. Troverete anche il circuito di alimentazione, il modulatore RF e vari cavi. Nel secondo capitolo è spiegato come connettere tutte queste parti. Occorre anche avere a disposizione uno schermo. Generalmente viene usato quello televisivo ma ne esiste anche uno appositamente realizzato per il computer - chiamato di solito monitor. Questo argomento verrà trattato in seguito. Tutti i componenti fisici vanno sotto il nome di hardware.

Avrete intuito da questa spiegazione che l'hardware non può fare molto da solo. Prima di poter realizzare giochi, musica o tenere la vostra contabilità sono necessari i programmi (il software). Un programma dice al computer ciò che deve fare. Senza di esso la macchina è inservibile!

Di fatto il computer possiede al suo interno un programma già al momento dell'acquisto. Esso diviene operativo quando accendete il sistema e fa in modo che la macchina possa capire quali tasti sono premuti sulla tastiera. Permette anche la realizzazione di operazioni comuni come far apparire messaggi o tracciare figure sullo schermo. In breve questo programma è alla base di alcune funzioni essenziali della macchina. Esso è anche chiamato programma monitor. Tale nome può generare confusione dal momento che abbiamo usato la parola monitor in riferimento allo speciale video del computer. Tuttavia, poichè generalmente viene usato lo schermo televisivo e noi non tratteremo dettagliatamente di questo pro-

gramma particolare, non si genereranno equivoci. Al limite quando vi imbatterete nei due termini saprete quale significato attribuirvi! Visto che il programma monitor è presente nella macchina e ne è sua parte integrante viene spesso chiamato firmware (dall'inglese firm = fisso).

Esso è situato nella parte del computer chiamata memoria. È sempre presente in quella particolare memoria chiamata ROM che permette di conservare permanentemente i programmi. In questo modo si evita che il programma vada perduto quando viene meno l'alimentazione. Invece i giochi e il software in genere sono tenuti in una parte della memoria dinamica detta RAM. Qualsiasi programma che volete usare è contenuto in essa indefinitamente ma scompare se lo sostituite con un altro o se togliete l'alimentazione al computer. Dove può essere il software che permette di realizzare i giochi o tutto ciò che è stato descritto all'inizio del capitolo? Il software si presenta sotto diverse forme, che possiamo catalogare secondo il modo in cui è memorizzato all'interno del VIC.

Tastiera

Se inventate un vostro programma, lo battete sulla tastiera e man mano viene automaticamente posto in memoria. Potrete trovare anche un buon numero di programmi pubblicati su riviste specializzate e libri. Dovete semplicemente copiarli battendone il testo alla tastiera. Quando avete finito, premete il tasto 'RETURN'e inizia il divertimento! Una volta terminato il libro sarete veramente abili alla tastiera. Dal momento che programmi brevi possono aiutare ad imparare l'uso del VIC, ne elencheremo alcuni che vi potranno divertire molto.

Nastro

Il VIC possiede anche uno speciale registratore a cassetta. Potete comprare il software su cassetta, pronto per essere caricato sul vostro VIC. Esistono dozzine, se non centinaia di programmi disponibili su nastro.

Avere a disposizione il software su cassetta ne rende più semplice e veloce il caricamento nel VIC. E non esistono errori di dattilografia! Potete usare il nastro anche per memorizzare programmi che avete tratto da riviste o libri o che avete creato da soli. Questa operazione è detta di salvataggio. Quando salvate un programma il contenuto della RAM è scritto su nastro. Ciò permette di poter conservare quanto scritto, per essere poi utilizzato in un altro momento.

Cartucce

Potete comprare i programmi in cartucce che sono poste in una particolare fessura nel VIC. In poche parole è come se il software fosse memorizzato in una ROM ed è presente non appena acceso il sistema. È perma-

nente e non può andare perduto. Le cartucce sono più costose delle cassette ma di uso più conveniente. Sono specialmente impiegate per conservare i programmi detti di servizio come un word processor o le utility per programmatori. Il programma occupa la sua ROM, lasciando la RAM del computer libera per immagazzinare i testi che desiderate elaborare o il programma che volete scrivere. Esistono cartucce che non contengono per nulla software, ma sono una espansione di memoria (RAM) per i vostri programmi più lunghi.

Dischi
Sono anche conosciuti con il nome di 'floppy disk' o 'dischetti'. Come il nastro delle cassette, memorizzano i programmi su supporto magnetico. È necessario connettere il drive al vostro VIC se usate i dischi. I drive sono piuttosto costosi ma hanno migliori prestazioni rispetto alle cassette. Sono più affidabili, hanno una maggiore capacità e le operazioni di caricamento e salvataggio sono più veloci. Se diverrete un giorno un vero programmatore, o userete il computer per affari (quando 'il tempo è denaro'), probabilmente acquisterete un drive.

Per iniziare

Sono necessarie poche nozioni per entrare nel mondo colorato del vostro amico computer. Le parti essenziali sono :

Il computer a colori VIC 20
Il contenitore delle cassette
L'alimentazione
Il modulatore RF
Il cavo del video (gli altri cavi sono già collegati)

Tutti questi componenti sono generalmente alimentati come normali dispositivi elettrici. È necessario anche un televisore, preferibilmente a colori, anche se il VIC può adattarsi anche ad uno schermo monocromatico. Per la musica e gli effetti sonori il VIC impiega il canale audio del televisore. Questa soluzione è diversa da quella utilizzata da altri computer che presentano al loro interno un altoparlante. Quindi il notevole vantaggio del sistema del VIC20 è una qualità del suono decisamente migliore e la possibilità di controllare perfettamente il volume. Potete tenerlo alto quando desiderate ascoltare o abbassarlo se state effettuando esperimenti di programmazione.

In alternativa al televisore potete impiegare un monitor, sia esso a colo-

ri o monocromatico. In questo caso otterreste figure più precise, ma non sarebbe possibile ascoltare gli effetti sonori del VIC a meno che non sia presente un canale audio. Pertanto prima di acquistare un monitor è importante consultare il rivenditore e assicurarsi che può essere collegato con il VIC.

La connessione al computer non è possibile per alcuni modelli recenti di televisori. Se avete qualche dubbio, consultate il vostro rivenditore al momento dell'acquisto per eventuali chiarimenti.

Assieme all'hardware, probabilmente vorrete comprare alcune cassette vergini per registrare i vostri programmi. Molti preferiscono usare nastri brevi affinchè la ricerca del software memorizzato sia più veloce. Sono disponibili cassette per computer a prezzi contenuti. Le lunghezze c10 e c15 sono quelle più adatte ai nostri scopi. È possibile usare anche le cassette musicali ma sono preferibili quelle specifiche per computer. Infatti i piccoli difetti del supporto magnetico possono avere un effetto irrilevante quando registrate suoni o voci ma disastroso se dovete scrivervi programmi. Ciò non causerebbe un effettivo danno al computer ma il software potrebbe non funzionare bene o addirittura non funzionare affatto. È pertanto preferibile usare cassette sulle quali è espressamente assicurata la possibilità di impiego col computer.

Sono anche in vendita cassette 'senza fondo'. Ciò significa che sono sprovviste di una zona non magnetizzata colorata posta al termine del nastro. Il vantaggio consiste nell'evitare di memorizzare sulla parte non magnetizzata quando si parte dall'inizio della cassetta. In caso contrario perdereste la prima parte del programma e dovreste ripetere l'operazione. Tuttavia non è difficile abituarsi ad iniziare la registrazione a partire da qualche pollice più avanti, quindi non è essenziale avere le cassette 'senza fondo'.

Il vostro amico VIC

Il VIC si presenta come un computer amico. Vi sono diversi computer che possono essere considerati in questo modo, ma che cosa rende il VIC speciale rispetto gli altri? Il fatto di avere una tastiera molto robusta e con tasti ben conformati. Essi sono ben evidenziati e potete rendervi conto della loro funzionalità. Tutte queste caratteristiche rendono più semplice entrare in confidenza con la macchina. La configurazione della tastiera è stata progettata in modo che la battitura dei programmi sia semplice ed esistono anche speciali tasti che possono essere usati da controllo per i giochi. Questi facilitano e semplificano l'uso della macchina.

Un ottimo aspetto della macchina è la visualizzazione sullo schermo. Senza alcun software aggiuntivo, il VIC produce sul video figure chiare,

ben evidenziate e ricche di colori. Tutti i messaggi che compaiono presentano caratteri grandi, in modo da poter essere letti facilmente. Possono essere anche in diversi colori per evidenziarne il significato. Il numero di colori utilizzati arriva ad un massimo di sedici, mentre il bordo ne può assumere otto. Ciò permette di ottenere, grazie anche alla vasta gamma di simboli grafici, immagini attraenti e stupefacenti, come vedremo nel capitolo tre.

Il componente del suono ha tre 'voci', cioè è in grado di emettere tre differenti note nello stesso tempo. L'altezza di ogni suono è controllata separatamente, in modo che la melodia può essere accompagnata sia da una parte di soprano sia di basso. Esiste anche un generatore di rumore in modo da poter simulare il rombo di un motore di aereoplano o il sibilo violento provocato dall'espansione rapida di un vapore ad alta pressione.

Combinando suoni e note in una rapida successione, cosa in realtà molto semplice (vedere capitolo cinque), potete produrre nuovi effetti. L'aggiunta di effetti sonori rende un programma più efficace e divertente. Ciò è un altro fattore che salderà l'amicizia con il vostro computer.

A questo punto il VIC e tutti gli altri componenti sono a vostra disposizione, pronti per essere usati. Passiamo al capitolo due per scoprire come connetterli assieme.

Capitolo 2
Inizializzazione del sistema

Il VIC possiede tutti i cavi necessari per collegare i componenti. Prima di connettere i cavi studiate la console del computer in modo da mettere in risalto le principali caratteristiche. Prima di tutto esaminiamo la tastiera. Nell'angolo superiore destro c'è una spia rossa. Questa si illumina quando accendete il computer. I connettori dei cavi sono sulla parte destra e posteriore della macchina. Sulla destra (fig. 2.1) abbiamo la presa di corrente per l'alimentazione. Su questa è scritto '9V' per ricordarsi che il cavo principale non vi deve mai essere collegato. Vicino a questo c'è l'interruttore per accendere il VIC. Alla sua sinistra c'è la porta di controllo. Vi potete collegare una infinità di cose, come i controlli dei giochi o la penna luminosa. Potrete collegarli al vostro sistema più avanti, se lo desidererete.

Il retro della macchina presenta cinque porte (fig. 2.2). Sulla sinistra (come potete vedere sulla parte posteriore del computer) c'è la porta per

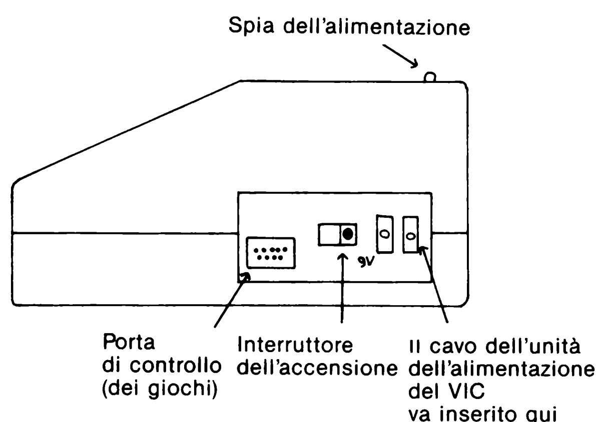


Fig. 2.1 - La parte destra della tastiera del VIC 20.

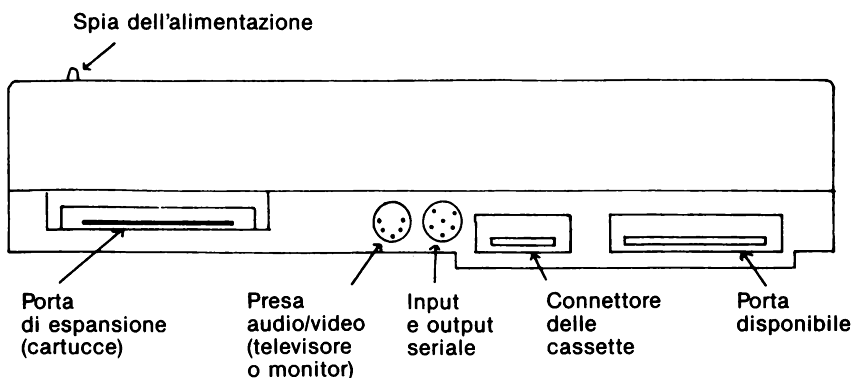


Fig. 2.2 - La parte posteriore della tastiera del VIC 20.

l'espansione della memoria dove potete connettere cartucce o schede per la memoria addizionale. Esse potranno essere utili in seguito, mentre non sono indispensabili agli inizi. Vediamo ora i due connettori circolari 'DIN'. Potete vedere che sono di forma diversa e quindi non è possibile collegare un cavo alla presa sbagliata. Ecco un altro esempio dell'amicizia del vostro computer! La presa audio/video è quella usata per connettere il modulatore che manda i segnali sonori e le immagini al televisore. Le altre prese circolari sono per il collegamento seriale di input/output. Noi non tratteremo tale argomento in questo libro. Brevemente, questa porta consente lo scambio di informazioni tra il computer e le altre periferiche. Per esempio se ponete un modem esso scambia dati con altri VIC20. Vi può essere collegata anche una stampante seriale.

La quarta porta sulla parte posteriore della macchina contiene il con-

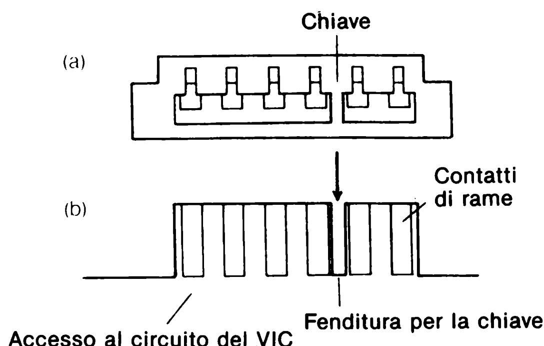


Fig. 2.3 - Connessione del registratore a cassetta al VIC 20. (a) Veduta della parte terminale del cavo del registratore. (b) Veduta dall'alto della piastra del circuito, così come appare dalla porta di ingresso del registratore.

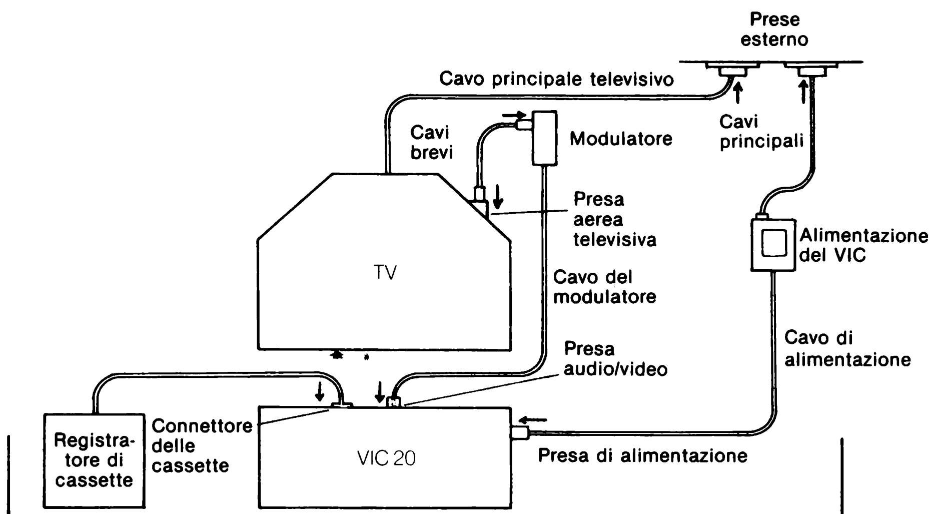


Fig. 2.4 - Come connettere le varie parti del sistema. Le frecce indicano in che modo i cavi devono essere inseriti nella presa.

nettore delle cassette. Si presenta con il bordo ampio e simile ad una lingua. Ha sei strisce di rame che chiamiamo connettori a pettine. Qui si collega il registratore a cassetta: il suo cavo ha infatti una presa a sei ingressi. Notate che esiste una fessura tra la seconda e la terza striscia di rame. La presa sul cavo della cassetta ha una chiave nella posizione corrispondente. Quando la spina è posta nella presa la chiave entra nella fessura. La presenza di questo meccanismo esclude la possibilità di eseguire il collegamento in modo errato.

All'estrema destra della superficie posteriore c'è un'altra porta disponibile. Qui possono esssere collegate altre periferiche come i controlli dei giochi, la penna luminosa, o una stampante.

Ponete ora il computer su di un tavolo, dove potete usarlo comodamente. Avrete bisogno di due prese principali, abbastanza vicine alla macchina. Una è per fornirle l'alimentazione, l'altra per il televisore. Quest'ultimo dovrà essere posto preferibilmente dietro il computer, oppure di fianco se le dimensioni del tavolo non lo consentono. Il prossimo passo è connettere assieme le parti del sistema. Il modulatore RF possiede un cavo lungo e uno corto (fig. 2.4). Il primo deve entrare nella porta video del VIC, il secondo nella presa aerea del televisore. Il cavo del registratore di cassette deve entrare nella porta delle cassette. Di fatto non userete il registratore fino a che non sarete arrivati alla fine del capitolo tre, quindi non è necessario collegarlo ora.

L'alimentazione ha due cavi. Uno di essi ha un connettore che entra

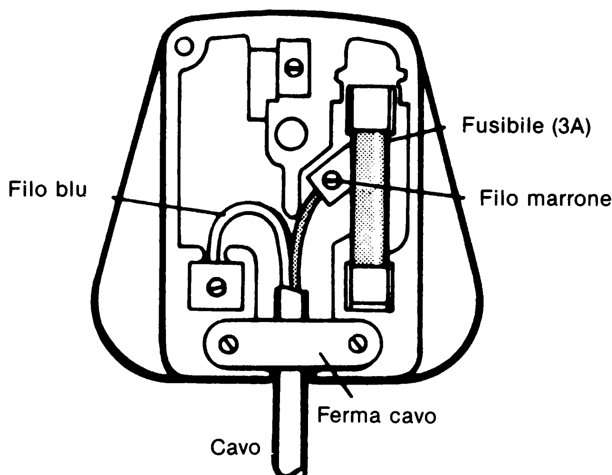


Fig. 2.5 - Connessione di una presa di 13 ampere.

nella presa situata sulla parte destra del VIC (fig. 2.1). Il secondo può essere già adattato con una spina. Se non lo è, fatelo voi stessi o chiedete l'aiuto di una persona esperta. La figura 2.5 mostra come collegare i fili in una spina standard a 13-ampère. Notate che i colori dei fili vi aiutano nella corretta esecuzione.

Connettete l'alimentazione e il televisore. Notate che la prima non ha alcuna luce di segnalazione così ricordatevi di togliere il cavo o spegnere l'interruttore esterno quando avete terminato l'uso del computer. Accendete la macchina mediante il piccolo interruttore posto sulla destra (fig. 2.1). La spia rossa che compare in cima al computer si illumina.

Accendete il televisore. Probabilmente non accade nulla perchè non è ancora sintonizzato con la macchina. Il modulatore del VIC è costruito in modo da inviare segnali al televisore sul canale 36. Se il vostro apparecchio è a bottoni sceglietene uno che non usate per i programmi. Premetelo e regolate la sintonizzazione fino a che non ottenete una immagine nitida e a colori come si vede nella fig. 2.6. Altri apparecchi televisivi differiscono nella selezione e sintonizzazione del canale, quindi consultate le istruzioni se avete qualche dubbio. Generalmente si impiegano poche decine di secondi prima di ottenere sullo schermo il messaggio nitido del VIC in funzione. Lo sfondo nel centro del video è bianco con lettere blu ed è circondato da un bordo in azzurro.

Sintonizzate il canale con precisione per evitare il più possibile immagini doppie e colori che sfumano sui contorni. Se potete riservare sempre lo stesso bottone del televisore, il tutto sarà pronto per l'uso del VIC ogni volta che lo accenderete.

```
***** CBM BASIC V2 *****
3583 BYTES FREE
READY.
■
```

Fig. 2.6 - Ciò che appare sullo schermo quando viene acceso il VIC 20.

Entriamo in confidenza con la tastiera

Se avete già usato una macchina da scrivere o un altro computer, probabilmente sapete dove trovare la maggior parte dei caratteri. In caso contrario sarete sorpresi da quanto velocemente imparerete. È stato detto che questa strana disposizione dei tasti (la tastiera QWERTY, come è spesso chiamata) sia stata creata con l'intento di rendere arduo il compito ai dattilografi! Il VIC risponde alla pressione di un tasto in una frazione di millisecondi, così che non sono possibili malfunzionamenti, anche se chi batte è velocissimo. Non dovete affatto preoccuparvi se premete il tasto sbagliato al momento sbagliato. Potete premere tutti i caratteri che volete in qualunque istante senza danneggiare il VIC. Naturalmente ciò provocherà strani effetti sullo schermo e sui programmi in memoria, ma questo è un altro discorso. Potete sempre riparare una simile situazione ripartendo da capo.

Tanto per iniziare, provate a premere un tasto qualsiasi. Scrivete il vostro nome, o uno dei vostri familiari, oppure ciò che vi suggerisce la fantasia. Se volete battere uno spazio, premete la barra che è sotto la tastiera. Tutto ciò che scrivete compare sullo schermo a lettere maiuscole, forse assieme a numeri e segni di punteggiatura se avrete sbagliato a battere! Se fate un errore, c'è un modo semplice per correggerlo. Premete solamente il tasto posto nell'angolo superiore destro della tastiera. Esso si presenta così:

INST
DEL

'DEL' è una forma abbreviata dell'inglese 'delete' (= cancellare). Il computer torna indietro di uno spazio e il carattere errato che avete appena scritto scompare dallo schermo. Potete fare un'altra scoperta circa questo

tasto. Se lo tenete premuto per qualche secondo o poco più, cancella molte lettere di seguito.

Tutte le volte che scrivete qualcosa, noterete un quadratino blu che 'lampeggia' regolarmente avanti e indietro. Esso è il cursore. È una specie di rivelatore della vostra posizione sullo schermo e vi mostra dove comparirà la prossima lettera che batterete. Appena premete un tasto, il cursore si sposta di una posizione a destra rispetto al carattere introdotto per ultimo. Alla fine della riga scivola sulla linea sottostante al primo spazio a sinistra. Quando premete il tasto 'delete' si muove a sinistra, cancellando le lettere sulle quali passa. È sorprendente ciò che riuscite a fare con il cursore! Provate ora il tasto che segue:



Esso è vicino a quello indicato con INST/DEL. Il suo effetto è di mandare il cursore alla sua 'casa', che è l'angolo in alto a sinistra dello schermo. A proposito, quando usiamo il termine schermo non intendiamo tutto lo schermo televisivo, ma solo la sua parte centrale dove appaiono i caratteri (che è bianca). Quella colorata in blu chiaro che appare tutt'attorno è chiamata bordo.

Ora userete i due tasti in cima all'angolo destro della tasiera per far correre il cursore per tutto il video. Essi sono entrambi chiamati CRSR, che è una forma abbreviata di 'cursore'. Il cursore si muove nella direzione mostrata dalle frecce. Premendo il tasto up/down si sposterà in basso, riga per riga. Come potete fare per muovervi in alto? Ottenete ciò mediante il tasto SHIFT. Cosa accade se premete lo SHIFT contemporaneamente a CRSR?

Ci sono due tasti SHIFT sul fondo. Ne sono necessari due perchè il loro uso è molto frequente. Non lo dovete mai usare da solo ma sempre assieme a qualche altro tasto. Il fatto che siano due rende agevole battere il carattere con una mano, mentre con l'altra si preme lo SHIFT. La tastiera è veramente amica di chi usa semplicemente due dita! Provate il tasto CRSR con e senza lo SHIFT e imparerete presto a guidare il cursore dove volete.

Probabilmente a questo punto lo schermo sarà pieno di una miriade di parole, lettere e figure. Ecco un altro uso del comando SHIFT. Premetelo assieme a CLR/HOME. Ora ci rendiamo conto di cosa significhi CLR: deriva da 'clear' ovvero pulire. Non solo porta il cursore in alto a sinistra, ma tutto il resto viene cancellato dal video. Ora avete a disposizione un'a-

rea bianca, pronta per iniziare da capo.

Ora premete lo SHIFT contemporaneamente agli altri tasti. Alcuni hanno un segno di punteggiatura al di sopra, come ad esempio:



Premendo il tasto da solo (cioè senza lo SHIFT) avrete come risultato il segno inferiore (/). Usando contemporaneamente lo SHIFT otterrete quello superiore (?), esattamente come una normale macchina da scrivere. Allo stesso modo i tasti dotati di una figura o di un simbolo di punteggiatura, per esempio:



danno il carattere (4) senza lo SHIFT e il simbolo ($) con lo SHIFT.

Potete ora capire come CLR/HOME mandi il cursore al'inizio se premuto da solo, ma cancella tutto ciò che è sullo schermo se usato assieme a SHIFT. Vi sono alcune eccezioni a questo. Battendo il tasto 'freccia-in-su' (↑) da solo esso fa comparire la freccia in su. Usato assieme a SHIFT produce invece la lettera greca 'pi' (π). Essa è segnata in fronte al tasto considerato, e non in cima come vi sareste aspettati.

Avete usato gli altri caratteri assieme a SHIFT? Se non lo avete fatto, provateci. La lettera 'A', per esempio, dà la 'A' maiuscola premuta da sola, ma assieme a SHIFT produce qualcosa di nuovo - un simbolo di 'picche' simile a quello delle carte da gioco. Avrete notato che esso compare di fronte al carattere 'A'. Infatti ci sono due caratteri di fronte a ogni lettera, così come ci sono sul '+', '—', '£' e '*'. Il simbolo che ottenete è sulla destra di ogni coppia. Li chiameremo 'gruppo della parte destra'. Ora vedremo quelli a sinistra.

Aiuto!

A questo punto potreste essere un pochino in difficoltà. Probabilmente sono apparse sullo schermo cose che non corrispondono perfettamente a quanto detto sino ad ora. Forse perchè avete premuto alcuni degli altri tasti di controllo al momento sbagliato, forse per errore o per sperimentare qualcosa di nuovo. Sarà anche apparso sul video il sinistro messaggio 'SYN-

TAX ERROR'(non è proprio amichevole come al solito!).

Non preoccupatevi! C'è un modo semplice per tornare a una situazione normale. Cercate il tasto RUN/STOP, che è sulla estrema sinistra e quello RESTORE, che è sulla parte opposta a destra. Premeteli assieme. Tutti i simboli, le lettere e i messaggi minacciosi saranno cancellati. L'incoraggiante parola READY (= pronto) apparirà nell'angolo superiore sinistro dello schermo, con il cursore che brilla tranquillamente vicino. Il VIC è ora nello stato in cui si trova subito dopo l'accensione. Se il computer nel frattempo avesse preso a scrivere in rosso o in altri colori, riprenderà ora a scrivere nel suo blu più familiare.

Spesso accade che non appaia nulla sullo schermo quando battete sulla tastiera. Questo accade perchè il colore nel quale scrivete è divenuto il bianco. Usare inchiostro bianco su carta dello stesso colore non è il modo più semplice per essere chiari! Come abbiamo spiegato prima, premendo RUN/STOP con RESTORE si ritorna di nuovo al blu.

I colori

Non è una cattiva idea giocare ancora con i tasti per qualche minuto. Provate a vedere come appaiono sul video tutti i simboli a destra. Forse potete usarli assieme per costruire dei disegni. La figura 2.7 mostra alcuni esempi. Potrebbe essere simpatico cambiare il solito noioso colore blu. Il VIC ha così tanti colori che è un peccato usarne uno solo. Avrete già trovato i nomi dei colori in fronte ai tasti dei numeri. Per cambiarli usate il tasto CTRL (controllo) che è alla estrema sinistra. Usatelo come lo SHIFT con uno dei tasti numerici. Provate con CTRL e il '3'. Questo porta scritto RED (= rosso) e, come ci aspettiamo, tutte le future lettere e simboli sul video saranno in rosso. Anche il cursore muta il colore, in modo che voi possiate sempre rendervi conto in quale colore state scrivendo.

Ora vediamo perchè le lettere e il cursore qualche volta scompaiono dal video. Premete il tasto CTRL e il '2', poi scrivete alcuni caratteri. Il cursore, come pure le lettere battute, è bianco e perciò risulta invisibile sullo sfondo anch'esso dello stesso colore. Esse esistono realmente, ma sono invisibili! Premete ora CTRL e il '6'. Il cursore riappare un po' più in giù nello schermo, provando che si muove veramente se scrivete in bianco. Ora avete scelto il verde. Il '9' e lo '0' non hanno colori, ma portano scritto 'RSV ON' e 'RSV OFF'. Per scoprire la loro funzione, premete il CTRL e il '9' (RVS ON) assieme. Ora battete poche lettere o simboli del gruppo di destra. Otterrete caratteri bianchi su sfondo verde. Per il bianco su fondo rosso, battete il CTRL con il '3'. Naturalmente, RVS ON è l'abbreviazione di 'reverse graphic on' (= grafica in reverse). Provate ora gli altri

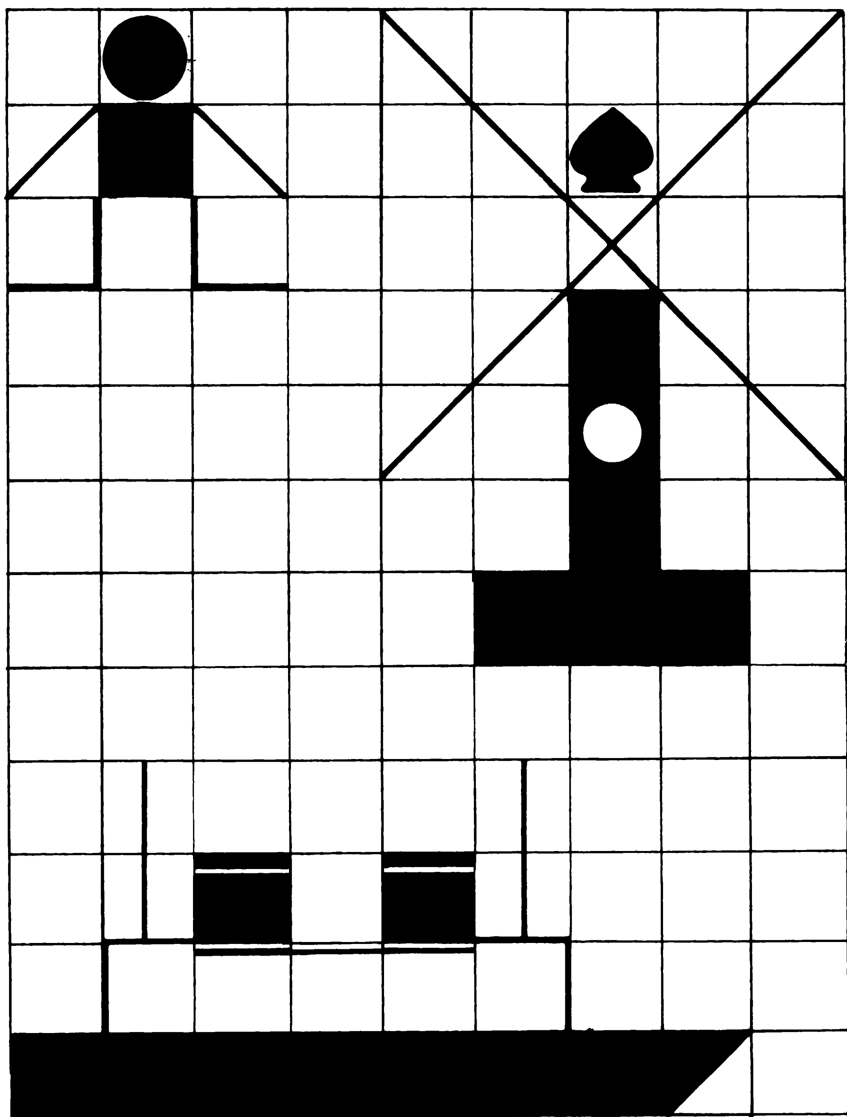


Fig. 2.7 - Figure create dai caratteri grafici di destra.

colori in reverse. Useremo questo tipo di scrittura più avanti, quando dovremo evidenziare titoli o parole importanti sullo schermo. Scrivere i caratteri in reverse raddoppia i modi a disposizione per creare figure interessanti. Quando vorrete ritornare ai caratteri normali, premete CTRL assieme a RVS OFF. Oppure, se lo preferite, potete usare RUN/STOP e RESTORE.

A destra e a sinistra.

Vi è un tasto alquanto strano nella estremità sinistra della tastiera. Mostra una ‘C’ parzialmente circondata da una bandierina. Essa è il simbolo commerciale del Commodore. La chiameremo semplicemente ‘bandiera’. Provate ad usarlo come lo SHIFT. Alla fine vi accorgerete che ciò produce il gruppo di simboli di sinistra. Per esempio:

Premendo il tasto ‘A’ senza SHIFT ottenette ‘A’

Premendo il tasto ‘A’ con lo SHIFT ottenete ‘♠’

Premendo il tasto ‘A’ con la ‘bandiera’ ottenete il simbolo ‘ ’

Scrivete alcuni caratteri della parte sinistra sullo schermo per rendervi conto di come essi appaino. Provateli in diversi colori e anche in reverse. Scrivete anche quelli di destra. Fate in modo di riempire il video con simboli di tutti i tipi, di destra e sinistra, lettere e numeri. Ora potete verificare la funzione di ‘cambiamento totale’ del VIC.

Questo effetto si verifica se fate una operazione di cui molto probabilmente vi siete chiesti il risultato: cosa accade se premete il tasto di SHIFT assieme alla ‘bandiera’? Forse lo avete già fatto: molti possiedono delle dita veramente intraprendenti! In caso contrario provate ora: premete il tasto SHIFT e la ‘bandierina’ e osservate sullo schermo cosa accade. In realtà succedono molte cose assieme. Per ritornare indietro, premete di nuovo SHIFT e la ‘bandiera’. Poi ripetete ancora l’operazione di ‘cambiamento totale’.

I suoi effetti sono:

	Esempio
Le lettere maiuscole vengono convertite in minuscolo	la A in a
I simboli di destra in lettere maiuscole	♠ in A
Alcuni simboli di destra in altri	in

I numeri, i segni di punteggiatura, quelli matematici non cambiano. E non mutano neppure alcuni caratteri di destra come i colori.

Ciò può sembrare a prima vista confuso ma l’idea che è alla base è semplice. Il VIC ha due modi per visualizzare i caratteri sullo schermo che sono chiamati modi. Uno è quello grafico, l’altro è quello di testo. Potete passare da uno all’altro battendo lo SHIFT assieme alla ‘bandiera’.

Quando il VIC è acceso per la prima volta (oppure lo avete inizializzato con RUN/STOP e RESTORE) lavora in modo grafico. Esso vi fornisce

le lettere maiuscole e i simboli di destra, quelli matematici e i numeri. Il modo grafico è preferibile se state battendo un programma.

Se ponete il VIC nel modo di testo, usando lo SHIFT e la 'bandiera', ottenete lettere maiuscole o minuscole secondo che usiate lo SHIFT oppure no. Questo metodo è il migliore se volete usare il VIC come una normale macchina da scrivere. Il modo di testo vi fornisce anche i simboli di sinistra (premendo la 'bandiera' da sola) che sono i più usati nella realizzazione di carte e tabelle.

La lavagna

Quanto stiamo per dire vi darà un completo controllo della tastiera e vi aiuterà a guidare il vostro talento artistico. Prima di tutto, battete:

```
POKE 36879,15
```

Poi premete RETURN. Il video diventerà nero con il bordo giallo. La lavagna è appunto lo schermo nero. Ora battete CTRL e '8' assieme. Il cursore cambierà da bianco in giallo e sarà il 'gesso'. Ora premete CTRL e il '9'. Ciò porrà il VIC in reverse, così che uno spazio compare come un quadratino giallo. Per cancellare la lavagna usate SHIFT con CTRL/HOME. Il cursore andrà nell'angolo in cima a sinistra dello schermo e tutto quanto scritto in precedenza scomparirà. Potete muovere il cursore in basso e a destra usando uno dei due tasti CRSR da soli mentre con lo SHIFT vi sposterete in alto e a sinistra.

Quando volete tracciare una linea con il gesso, posizionatevi alla sinistra del punto da cui volete farla partire e premete la barra. Premuta una volta produrrà un singolo quadratino giallo, mentre se la tenete abbassata disegnerete una linea orizzontale composta da parecchi quadrati. Poi muovete il cursore ancora dove volete tracciare un'altra riga e usate la barra. Per disegnare linee verticali occorre farlo quadrato per quadrato, muovendosi in basso e a sinistra.

Cambiate il colore del gesso con il tasto CTRL e uno dei numeri da 1 a 8. Per cancellare un qualsiasi quadrato disegnato, posizionatevi alla sua destra e battete INST/DEL. Poi premete INST/DEL con SHIFT e potrete ripartire da capo in ogni istante con SHIFT e CLR/HOME. Buon divertimento con i vostri scarabocchi!

Tasto dopo tasto

Tutto questo necessita di un certo esercizio, ma dopo un po' di pratica di-

venta tutto molto naturale. A questo punto non è necessario imparare la funzione di tutti i tasti, ma usando il VIC vi renderete conto dei tasti che impiegate più frequentemente e del loro significato. Quando avrete acquistato una maggiore familiarità con la macchina imparerete anche ad usare i meno comuni.

Le prime quattro tabelle che potete consultare nell'Appendice A vi possono aiutare ogni volta che siete in difficoltà. Dalla prima alla terza elencano tutti i tasti e le loro funzioni mentre la quarta elenca le altre possibilità di utilizzo della tastiera. Quando volete fare qualcosa ma non vi ricordate come, consultate questa tabella e il tasto che dovete premere. Probabilmente ciò accadrà all'inizio, ma poi ne farete a meno man mano che diverrete esperti del VIC.

Capitolo tre
Una fantasia di colori

Se non vedete l'ora di ammirare i colori sfavillare sullo schermo, saltate direttamente al programma listato in 3.3. Ma se seguirete questo capitolo passo dopo passo imparerete molto presto come ottenere dal VIC i suoi migliori colori.

Accendete il VIC e premete il tasto CLEAR/HOME con SHIFT per cancellare lo schermo e mandare il cursore in alto a sinistra. Per ottenere un effetto migliore dai colori è preferibile usare uno sfondo adatto. Una delle cose più semplici da fare sul VIC è cambiare il colore dello schermo e del bordo che lo circonda. Uno dei comandi da dare è il seguente:

```
POKE 36879,207
```

Provatelo esattamente come è scritto avendo cura di non sbagliare i numeri. Appena avete battuto il comando, esso appare sul video ma non accade nulla. Il VIC sta solamente ricevendo ciò che scrivete sullo schermo. Ora premete RETURN: l'effetto è istantaneo. Il colore dello sfondo cambia da bianco a porpora chiaro, mentre il bordo passa da azzurro a giallo.

Il comando POKE è molto usato, quindi vediamo cosa significa. Come descritto nel capitolo uno, il VIC possiede una memoria di cui potete cambiare il contenuto, chiamata RAM. RAM è una abbreviazione di 'Random Access Memory' (= memoria ad accesso casuale). Il concetto fondamentale circa la RAM è che qualsiasi informazione ivi memorizzata vi rimane finchè non la cambiate o non spegnete la macchina. Fisicamente la RAM consiste in un insieme di circuiti elettronici, tuttavia pensatela come un insieme di piccole scatole ognuna delle quali possiede una fessura sul coperchio (fig. 3.1). Ciascuna scatola ha un numero che la identifica ed è chiamato indirizzo. L'indirizzo di quella che abbiamo appena usato è 36879. Questa è una speciale scatola (cella di memoria) destinata a ricevere le istruzioni circa il colore che devono assumere lo schermo e il bordo. Quando battete 'POKE 36879,207' è come se scriveste il numero '207'

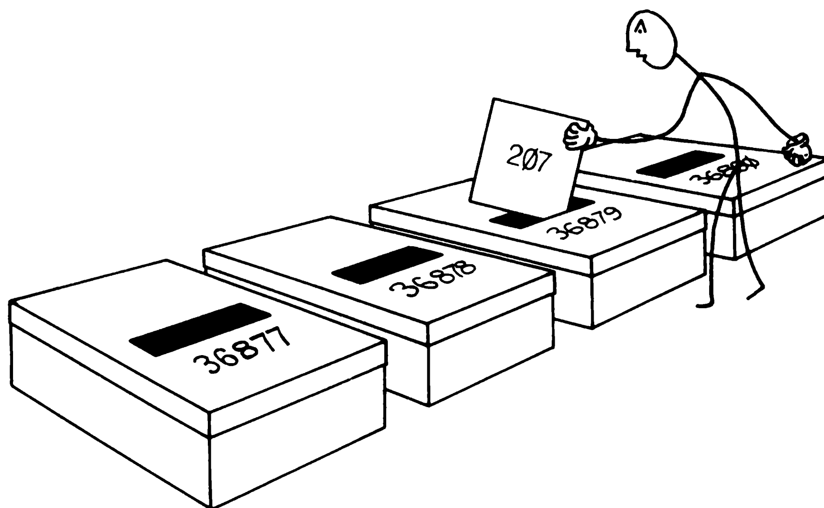


Fig. 3.1 - Attraverso l'operazione di POKE il valore 207 è introdotto nella RAM all'indirizzo 36879.

su di un pezzo di carta e lo introduceste nella fessura della scatola. Quando state introducendo numeri nella RAM mediante l'istruzione POKE non preoccupatevi di poter per errore scrivere sulla ROM. L'istruzione POKE non può aver alcun effetto sulla ROM.

Quando scrivete 'POKE 36879,207' il VIC visualizza sullo schermo il vostro comando, ma non lo interpreta. Solo quando premete RETURN è come se diceste al VIC: 'ecco, questo è ciò che voglio che tu faccia. Va ed esegui!'. Il VIC fa esattamente ciò che gli è stato ordinato: prende il numero 207 e lo mette nell'indirizzo 36879 nella sua memoria. Non appena ciò è stato eseguito, la parte del VIC che controlla il video trova questo nuovo numero all'indirizzo 36879 e cambia il colore dello schermo e del bordo come ordinato.

I numeri che possono essere impiegati vanno da 0 a 255, anche se quelli che producono qualche effetto appartengono a un intervallo più limitato (come già detto da 1 a 8). Ciò dipende dall'indirizzo interessato. La Tabella 5 (vedere Appendice A) mostra i numeri consentiti per l'indirizzo 36879 e il loro effetto sui colori del video e del bordo. Provatene alcuni: scrivete 'POKE 36879' seguito da uno dei numeri della Tabella 5, poi premete RETURN. Non dimenticate la virgola tra i due numeri. In caso contrario il VIC non capisce ciò che intendete e risponde con '? ILLEGAL QUANTITY ERROR' (= errore per quantità illegale). Poi visualizza 'READY' (= pronto) per farvi sapere che potete riprovare. Se per caso fate degli errori scrivendo questi numeri, usate il tasto INST/DEL per can-

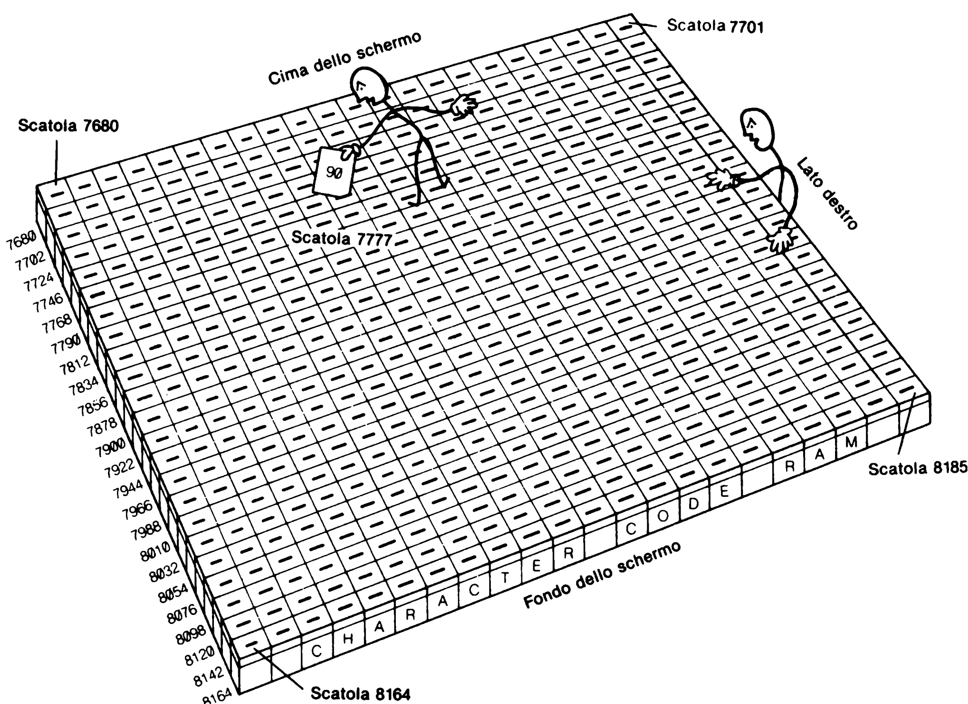


Fig. 3.2 - Attraverso l'operazione di POKE viene introdotto nella RAM dei codici dei caratteri il codice per un 'rombo', all'indirizzo 7777. (Nota: Gli indirizzi sono diversi se il VIC possiede più di 3K di RAM aggiuntiva; vedere l'Appendice B).

cellare i caratteri errati, come descritto nel capitolo due. Poi scrivete la giusta configurazione.

Ora iniziamo a disegnare (o a stampare come viene più spesso detto) alcune immagini sullo schermo. Battete lo stesso comando per ritornare allo sfondo porpora con il bordo giallo. Cancellate il video e posizionate il cursore in alto a sinistra. Ora battete:

```
POKE 7777,90
```

Premete RETURN. Notate come non accada nulla ma ora c'è una immagine di un rombo bianco.

Come prima, avete scritto la parola 'POKE' seguita da due numeri. Il primo è l'indirizzo nella RAM. Il VIC ha una parte della memoria riservata alle informazioni riguardanti lo schermo. Ci possono essere 23 righe di caratteri sul video e ciascuna riga ne contiene 22, quindi in totale è possibile avere 22*23=506 caratteri sul video. La RAM dello schermo

necessita di 506 indirizzi, uno per ciascun carattere. Questa è la RAM per i codici dei caratteri. Come mostra la figura 3.2, l'indirizzo corrispondente all'angolo sinistro è 7680. Quello in basso a destra è 8185. Il diagramma mostra che appare un rombo bianco se si usa l'istruzione POKE all'indirizzo 7777. Provatene degli altri, per esempio:

POKE 7778,81

Premete 'RETURN' e vedrete apparire un disco bianco vicino al rombo. È chiaro che il secondo numero che ponete nella cella mediante l'istruzione POKE indica al VIC quale forma o quale altro carattere debba essere stampato. Questo è il codice del carattere. La tabella 6 vi mostra quale numero usare per ottenere le lettere o i simboli che desiderate. Provatene alcuni per entrare in confidenza con con il metodo.

Quando scrivete sull'ultima linea dello schermo e premete RETURN, tutto ciò che è scritto si sposta di una linea verso l'alto. Questo processo è chiamato scrolling. Normalmente ciò non crea difficoltà, ma se per esempio avete usato l'istruzione POKE nella cella 7777 per far comparire un rombo, il computer prende il rombo dalla cella 7777 e la mette nella 7755, così che è visualizzato una linea più in alto dello schermo (da 7777+30720 a 7755+30720). Se ora voi volete alterare il colore del rombo, non dovete più usare nell'istruzione POKE il numero 7777+30720. Dopo che lo scrolling si è verificato più volte, probabilmente avrete perso le tracce di tutti gli indirizzi.

Per evitare questo inconveniente, fate in modo che il cursore non raggiunga l'ultima linea. Prima di iniziare ogni sequenza di comandi, cancellate lo schermo e mandate il cursore in alto a sinistra (CLR/HOME con SHIFT). Ciò vi dà abbastanza linee per scrivere tutto ciò che volete.

Se volete evitare di scrivere ogni volta 'POKE', usate la forma abbreviata di questo comando. Il VIC accetta le versioni abbreviate di tutti i comandi e tasti (vedere l'Appendice D di Personal Computing on the VIC 20). Vale la pena di ricordare quelle che userete spesso. Invece di POKE battete 'P' e poi SHIFT e 'O' assieme. Sullo schermo vedrete la 'P' seguita dal simbolo sulla destra del tasto 'O'. Se già avete nella giusta posizione

il disco e il rombo, siete pronti per proseguire. In caso contrario cancellate lo schermo e ripetete le due istruzioni POKE già date. Le due figure sono bianche perchè non avete dato al VIC un altro colore. Potete fare ciò mediante un altro indirizzo. Il VIC ha un secondo gruppo di 506 indirizzi nei quali scrivere le istruzioni concernenti i colori. Essi vanno da 38400 a 38905 (vedere fig. 3.3) e costituiscono la RAM dei codici di colori. La sequenza di indirizzi in questa RAM coincide esattamente con quella della ROM corrispondente. A ciascun indirizzo della RAM ne corrisponde uno nella ROM. La differenza tra i due è esattamente 30720. Per cambiare il colore del rombo, prima calcoliamone l'indirizzo del codice dei colori; 7777+30720=38497.

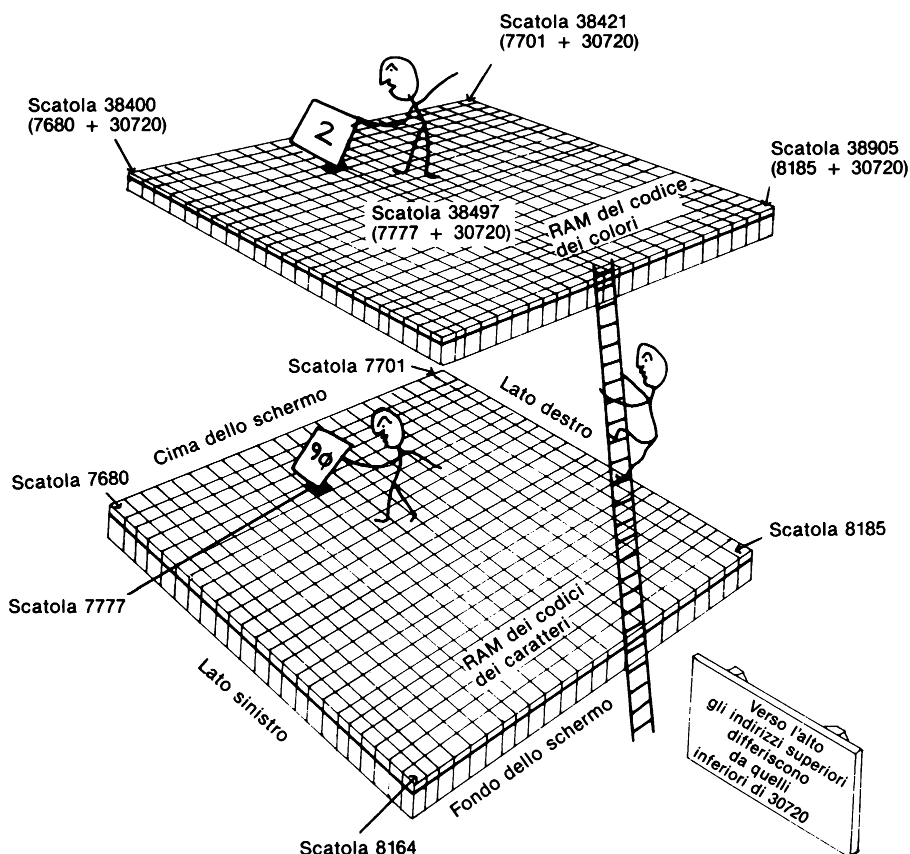


Fig. 3.3 - Attraverso l'operazione di POKE viene introdotto nella RAM dei codici dei colori all'indirizzo 38497 il codice del 'rosso', per colorare il 'rombo' che è stato memorizzato nella RAM dei codici dei caratteri. (Nota: Gli indirizzi sono diversi se il VIC possiede più di 3K di RAM aggiuntiva; vedere l'Appendice B).

Ora battete:

POKE 38497, 2

Il rombo diventa rosso. (Vi siete ricordati di premere RETURN? Dovete sempre farlo se volete che il VIC entri in funzione.) Il '2' è il codice del colore rosso. La tavola 7 dell'Appendice A mostra gli otto codici dei colori possibili. Ora provate a cambiare il disco bianco in uno verde. Prima di tutto calcolate 7778+30720=38498. Il codice per il colore verde è '5'. Poi battete:

POKE 38498,5

e il disco diventa verde. Fate qualche esperimento sullo schermo, cambiando il colore di questi e di altri caratteri. Provate a mettere tre dischi (codice 81) uno vicino all'altro con il primo all'indirizzo 7787. Ci sono 22 caratteri su ciascuna linea, pertanto gli indirizzi degli altri due sono 7787 + 22=7809, e 7809+22=7831. Poi fateli diventare rossi, gialli e verdi come a un semaforo stradale.

Siete stanchi di calcolare le somme tra numeri così lunghi? Perchè non lasciate che sia il VIC a farle per voi? Invece di dire al VIC 'POKE 7809,81' battete solamente:

POKE 7787 + 22,81

Il VIC farà la somma prima di eseguire l'istruzione. Allo stesso modo potete cambiare il colore del disco, lasciando che sia il VIC ad aggiungere anche 30720:

POKE 7787 + 22 + 30720,7

Questo comporta una scrittura più lunga, ma meno calcoli.

Il '+' e il tasto SHIFT

In altri computer dovete battere il tasto SHIFT per scrivere il '+'. Questa può divenire una abitudine difficile da lasciare. Se usate lo SHIFT con il '+' sul VIC, otterrete un segno grafico che assomiglia molto al simbolo di addizione. Leggendo quello che avete scritto potreste non notare l'errore, ma il computer non accetta questo carattere e ciò può generare altri inconvenienti in seguito. Così quando volete che la macchina esegua la somma, battete il '+' senza il tasto SHIFT.

Un programma per i colori

Il disegno della configurazione del semaforo stradale può essere ottenuto mediante sei comandi, ognuno seguito da RETURN. Dapprima i dischi appaiono bianchi, poi uno dopo l'altro diventano colorati. Invece di fornire alla macchina un'istruzione per volta, potete elencarle in un'unica lista e fare in modo che vengano eseguite in rapida successione. Il VIC sarà così veloce che vi sembrerà di aver eseguito tutto ciò istantaneamente, anche se non è di fatto così. Quando gli fornite una piccola sequenza di istruzioni, avete creato un programma. Un programma per un semaforo è mostrato nella lista 3.1.

```
10 POKE 7787,81
20 POKE 7787+22,81
30 POKE 7787+44,81
40 POKE 7787+30720,2
50 POKE 7787+22+30720,7
60 POKE 7787+44+30720,5
```

Listato 3.1 - Un programma di un semaforo.

Il VIC è stato veloce nell'eseguire le somme! Il programma ha un numero all'inizio di ogni riga (il numero di linea). Scrivete il programma e battete RETURN al termine di ciascuna linea. Il computer non agisce su ogni istruzione immediatamente. Il numero di linea all'inizio gli dice che quella è una linea di un programma. Questo fatto è memorizzato (registrato nella memoria) ma non è utilizzato immediatamente.

Se fate degli errori di scrittura, è meglio premere RETURN e ripetere la linea di nuovo (non dimenticatene il numero). Il VIC dimentica la riga precedentemente introdotta e ricorda solo l'ultima. Quando le avete scritte tutte e sei verificate ancora se avete commesso qualche errore e se li trovate ripetete la riga. Non importa se dovete ribattere le prime non appena avete finito le ultime. Le linee sono ordinate secondo il loro numero. Se avete dovuto scrivere di nuovo qualche istruzione e desiderate vedere il risultato finale nel giusto ordine, battete LIST seguito da RETURN e così facendo visualizzate l'intero programma. Questo ora è corretto e presente nella memoria del VIC. Per poterlo eseguire battete RUN. Non accade nulla, ma quando battete RETURN il programma funziona esattamente come previsto. In un istante appariranno sullo schermo i tre dischi nei loro giusti colori. È accaduto così velocemente che non avete avuto neppure il tempo di accorgervi che erano inizialmente bianchi, prima che mutassero colore. Se desiderate rivederlo, cancellate lo schermo (SHIFT e CLR/HOME) e battete RUN seguito da RETURN. Il programma funziona come voluto. Potete anche cambiarlo per ottenere colori o figure diverse. Ripetete ogni linea per sostituire quelle nuove a quelle vecchie.

Il Tappeto Magico

Il Tappeto Magico è un programma breve, facile da capire ma produce una fantasia di colori molto eccitante. Il miglior modo per comprendere come funziona è costruirlo passo dopo passo. Ciò sigifica introdurre le linee senza preoccuparsi del loro ordine perchè, come già spiegato prima, questo non è essenziale per il funzionamento del VIC. Se avete eseguito le istruzioni del paragrafo precedente, il vecchio programma sarà ancora in memoria e il video sarà coperto da parole e simboli. Per essere pronti ad introdurre il nuovo programma battete prima NEW e poi RETURN (in questo modo ci si libera del vecchio) e quindi RUN/STOP e RESTORE per riportare lo schermo nel suo stato immediatamente successivo alla inizializzazione. Iniziate a battere le linee dalla 50 alla 80 (vedere il listato 3.2).

```
50 FOR J = 7680 TO 7701
60 POKE J,81
70 POKE J+30720,2
80 NEXT J
```

Listato 3.2 - Il primo passo verso il programma del Tappeto Magico.

Lanciate il programma (battete RUN e poi RETURN) e vedrete apparire una fila di dischi rossi in cima allo schermo. Le due linee al centro sono semplici da comprendere. La numero 60 pone mediante un'istruzione POKE il carattere di un disco all'interno dell'indirizzo il cui numero è J. J può assumere diversi valori, come sarà spiegato più avanti. La linea 70 fa uso di un indirizzo che è pari a J, incremetato di 30720. Se J è l'indirizzo nella RAM dei codici dei caratteri, J + 30720 è quello nella RAM dei codici dei colori. Il '2' sta ad indicare che il disco è rosso. Le linee interessanti sono la 50 e la 80. La 50 fa sì che il VIC esegua un'operazione (i due POKE) più volte. La prima volta il valore di J è 7680 che è l'indirizzo dell'angolo sinistro dello schermo (vedere fig. 3.2). Alla linea 60 viene visualizzato il disco e alla 70 lo si pone di colore rosso. Alla 80 il comando 'NEXT J' produce un incremento di J pari a 1, per cui il nuovo valore è 7681. Il VIC ritorna alla linea 80 e ripete l'operazione, solo che questa volta J è 7681 e il disco appare al secondo posto della prima riga dello schermo. Si ripete il processo con J che assume via via i valori di 7682, 7683, 7684, fino ad arrivare a 7701 (la linea 50 dice infatti 'FOR J=7680 TO 7701' ovvero 'per J da 7680 a 7701'). L'ultimo valore corrisponde alla posizione più a destra della prima riga. A questo punto il VIC ha stampato una linea intera di dischi. J ha così raggiunto il suo massimo valore ed il programma ha termine e compare la scritta 'READY' (= pronto) sulla riga successiva. Sarebbe buona cosa saper creare dischi di tutti

i colori, scelti a caso in modo che voi non sapete di quale colore sarà la figura che compare ogni volta che lanciate il programma. Per realizzare ciò è necessario aggiungere una linea del tipo:

```
70 POKE J + 30720,C
```

nella quale C ha un valore qualsiasi compreso tra 0 e 7,in modo da produrre tutti i colori elencati nella tabella 7 dell'Appendice A.
Si può raggiungere questo scopo mediante una funzione BASIC chiamata 'RND', che è una forma abbreviata per 'random' che significa a caso. Se usate l'istruzione 'C=RND(1)', il VIC darà a C un valore compreso tra 0 e 1. Provate a verificarlo — ciò non fa parte del programma — battendo solamente:

```
PRINT RND(1)
```

e premendo RETURN. Verrà stampato un numero di dieci cifre, preceduto dal punto decimale. Otterrete un risultato diverso ogni volta che ripeterete l'operazione e non c'è alcun modo per prevedere il numero generato. Esso potrà andare da 0.000 000 0000 a 0.999 999 9999 e avreste bisogno di molto tempo prima di ottenere di nuovo lo stesso numero! Però non può essere usato come un codice di colore perchè non è un intero compreso tra 0 e 7. Ora provate questo:

```
PRINT RND(1)*8
```

In questo modo generate un numero compreso tra 0 e (quasi) 8. La parte dopo il punto decimale è troncata e 'dimenticata' usando un'altra funzione chiamata 'INT' (abbreviazione di 'integer'=intero), come segue:

```
PRINT INT(RND(1)*8)
```

Attenzione alle parentesi! È norma generale che, in ogni espressione, il numero di parentesi verso destra sia uguale a quello delle parentesi verso sinistra. Provate il comando più volte e vi renderete conto che genera numeri interi che vanno da 0 a 7. Il simbolo '*' significa 'moltiplica'. Il numero casuale tra 0 e 0.999... è moltiplicato per 8 per dare un risultato compreso tra 0 e 7.9999... Non otterrete mai 8 perchè mediante la generazione casuale si ha sempre un numero minore di 1.

Ora ponete tutto ciò nel programma. Battete una nuova linea:

```
20 C = INT(RND(1)*8)
```

e cambiate la numero 70 in:

```
70 POKE J + 30720,C
```

Lanciate ripetutamente il programma e ogni volta il valore di C e conseguentemente il colore dei dischi sarà casuale. Qualche volta potrete ottenere C = 0 corrispondente a una linea di dischi bianchi, invisibili su di uno schermo di egual colore. Potete renderli visibili mediante l'istruzione POKE 36879 (vedere la Tabella 5 dell'Appendice A). Se RUN e READY appaiono sull'ultima linea cancellate lo schermo e ripartite dall'inizio.

Se avete perduto delle linee usate LIST

Quando state scrivendo un programma, specialmente se fate molte correzioni, aggiungendo nuove linee o riscrivendone delle vecchie, è possibile che vi dimentichiate esattamente ciò che avete scritto. Potete vedere l'intero programma in ogni istante battendo LIST, poi premendo RETURN. Fate questa operazione molto spesso: vi aiuta ad avere una chiara idea del programma che state introducendo.

Ed ora miglioriamo leggermente il programma. Così come potete ottenere colori a caso, vediamo di stampare su linee casuali sullo schermo. Invece di stampare sulla prima riga, il VIC ne sceglierà una caso. Poichè ci sono 23 linee, numerate da 0 a 22, il VIC necessita di numeri casuali tra 0 e 22. Potete usare lo stesso comando:

```
X = INT(RND(1)*23)
```

L'espressione genera un numero tra 0 e 22, uno per ciascuna linea. Potete vedere l'indirizzo della posizione a sinistra di ogni linea nella Fig. 3.2. L'indirizzo di ciascuna linea si trova aggiungendo 22 a 7680. Ad esempio quello della riga X è 7680 + X*22. Per trovare l'ultimo indirizzo in una linea dovete calcolare 7701 + X*22. Introducete questa istruzione che sceglie un valore per la X:

```
30 X = INT(RND(1)*23)
```

Poi cambiate la linea 50 come segue:

```
50 FOR J = 7680 + X*22 TO 7701 + X*22
```

Lanciate più volte il programma e ottenete delle righe sullo schermo a diversi livelli e colori.

Può essere molto fastidioso continuare a battere RUN e cancellare tutti i RUN e READY che via via compaiono sullo schermo. Per far ripetere il programma al VIC indefinitamente, scrivete la linea:

```
150 GOTO 20
```

Generalmente GOTO si scrive in una parola sola, anche se è possibile scinderla in due. Ora lanciate il programma e vedrete che quando arriva alla fine, ricomincia da capo. Esso continua e continua. Gradualmente ciò che avete scritto sarà cancellato da righe di dischi colorati. Quando volete fermarlo, premete il tasto RUN/STOP.

E ora che ne dite di cambiare anche il colore dello schermo e del bordo? La generazione di numeri casuali sembra un metodo ideale per raggiungere questo scopo. La Tabella 5 mostra che occorrono numeri tra 8 e 255. L'espressione:

```
INT(RND(1)*248)
```

fornisce numeri tra 0 e 247, cosicchè aggiungete 8 e avrete numeri tra 8 e 255. Introducete questa nuova linea:

```
40 POKE 36879,8 + INT(RND(1)*248)
```

L'indirizzo 36879 è già stato menzionato precedemente in questo paragrafo. Lanciate il programma: ora i colori stanno veramente chiazzando lo schermo! Un aspetto che impoverisce l'effetto ottenuto è che tutto ciò che è presente sul video precedentemente all'inizio dell'esecuzione non è cancellato fino a che è coperto da un disco. Per ottenere ciò facciamo una cosa che può apparire strana a chi è abituato ad altre macchine. Le istruzioni sono date al VIC sotto forma di messaggi. Possiamo pensare di mandarne uno che non appare sul video. Introducete la linea 10 all'inizio del programma:

```
10 PRINT‘
```

Ora premete SHIFT e CLR/HOME assieme, come se doveste cancellare lo schermo. Il video rimane inalterato, ma compare dopo le virgolette un simbolo di un cuore in reverse. Battete le virgolette per chiudere la frase. La linea 10 ora appare così:

```
10 PRINT ‘♥’
```

Avete incluso un Codice di Controllo nel messaggio. Esso non compare mai sullo schermo durante l'esecuzione, ma viene letto e interpretato come un comando per cancellare lo schermo. Lanciate il programma e osservate ciò che succede. Ci sono 23552 possibili configurazioni, perciò visualizzandone al ritmo di 60 al minuto in due ore e mezzo le avrete viste tutte!

Ecco un programma completo che crea non solo linee ma anche colonne di dischi alternativamente. La seconda parte (dalle istruzioni 90 a 140) è molto simile alla prima. Non preoccupatevi circa il significato di 'STEP 22' nella linea 110, sarà spiegato nel capitolo sei. Introducete le linee dalla 90 alla 140 e lanciate il programma mostrato nel listato 3.3.

```
10 PRINT"♥"
20 C = INT(RND(1) * 8)
30 X = INT(RND(1) * 23)
40 POKE 36879,8 + INT(RND(1) * 248)
50 FOR J = 7680+X*22 TO 7701+X*22
60 POKE J,81
70 POKE J + 30720,C
80 NEXT J
90 C = INT(RND(1) * 8)
100 Y = INT(RND(1) * 22)
110 FORJ = 7680+Y TO 8164+Y STEP 22
120 POKE J,81
130 POKEJ + 30720,C
140 NEXTJ
150 GOTO 20
```

Listato 3.3 - Il Tappeto Magico.

Bene! Alcuni lettori potranno obiettare che i tappeti magici sono generalmente di origine orientale. I colori e le figure sono troppo rozzi e arditi per provenire dalla lampada di Aladino. Nessun problema! Mentre il programma è in esecuzione, premete SHIFT e il tasto 'bandiera' assieme. Ciò pone il VIC nel modo di testo. Il programma funziona come prima, solo che avete delle 'Q' invece dei dischi. L'effetto delicato è quello che avreste nel disegno di un tappeto persiano.

Riassunto

In questo capitolo vi è stato spiegato come:

- cambiare il colore dello schermo e del bordo
- fare stampare dal VIC qualunque carattere (lettere, numeri o simboli)
- fare stampare al VIC qualunque carattere negli otto colori
- introdurre un programma e farlo eseguire
- usare la forma abbreviata di alcune espressioni di BASIC
- fare eseguire le somme al VIC

— generare numeri casuali appartenenti a qualsiasi intervallo

Avete anche imparato:

— a premere RETURN per far sì che il VIC accetti i comandi
— a premere RUN/STOP per fermare un programma
— che PRINT visualizza i caratteri sullo schermo
— che POKE X,Y, pone il valore di Y nell'indirizzo X della memoria
— che NEW cancella i programmi presenti in memoria
— che FOR X = Y TO Z fa ripetere al VIC la stessa parte del programma più volte. X è all'inizio pari a Y poi viene incrementato di 1 fino a raggiungere il valore di Z
— che la funzione RND(1) genera numeri casuali da 0 a 1 (escluso)
— che la funzione INT tronca le cifre alla destra della virgola decimale ottenendo un intero. Nota: ciò è possibile solo con numeri positivi (come sarà spiegato in seguito)
— che GOTO fa ritornare il VIC indietro alla linea il cui numero compare nell'istruzione GOTO

Ormai quasi tutti voi vi sarete resi conto di quanto sia divertente programmare!

Capitolo 4
Cenni di BASIC

Prima di scoprire i divertenti effetti sonori che il VIC è in grado di realizzare, analizziamo più da vicino in che modo il computer elabora i numeri, una funzione questa alla quale sono finalizzati tutti i calcolatori. Accendete la macchina e datele il compito di eseguire la funzione aritmetica fondamentale:

```
PRINT 2 + 2
```

Come potete aspettarvi, la risposta, 4, è immediatamente visualizzata non appena premete RETURN. A proposito non dimenticate l'avvertenza circa il '+' e il tasto SHIFT nel capitolo tre. Dal momento che useremo molti PRINT in questo capitolo, è buona cosa impararne la forma abbreviata. Come tanti altri computer il VIC accetta il carattere '?' invece di PRINT. Userete la forma più lunga nei programmi, mentre sul video potete scrivere:

```
? 2 + 2
```

ottenendo come risultato 4. Naturalmente non compare nulla fino a che non premete RETURN. Ma questa probabilmente sarà già diventata un'abitudine. Da questo punto in avanti non vi rammenteremo più di battere RETURN dopo ogni comando.

Provate qualche altra addizione, come per esempio:

```
PRINT 2 + 4 + 6
PRINT 2 + 4 + 6 + 8 + 10
PRINT 2.5 + 4.7
```

Come potete vedere, i numeri decimali sono trattati come interi. Per battere il punto decimale usate il punto di fine periodo. Potete sommare numeri di nove cifre al massimo:

```
PRINT 1.23456789 + 9.87654321
```

Provate questo esempio e noterete che anche il risultato ha nove cifre.

Operatori

Il segno '+' è chiamato operatore. Posto tra due numeri produce una operazione su di essi. Quella causata dal '+' è l'addizione. Provate gli effetti dei seguenti operatori:

```
PRINT 6 — 4
PRINT 6  * 4
PRINT 6  / 4
```

Le risposte vi assicureranno che '—' è l'operatore della sottrazione, '*' della moltiplicazione e '/' della divisione.

Ora introducete:

```
PRINT 4 / 6
```

Questa divisione produce un risultato con infinite cifre decimali. Esso sarebbe realmente '0.666 666 666 666 666 666...7 e così via senza fine. La macchina non può visualizzare più di nove cifre, quindi la risposta è:

```
.666666667
```

Notate che se le cifre dopo il punto decimale sono zero, non appare nessun numero. La stessa regola vale quando introducete un numero. Se volete scrivere 0.456, ad esempio, non c'è alcun bisogno di mettere lo zero; battete solamente '.456'. Notate anche che l'ultima cifra è stata arrotondata per eccesso. Se chiedete al computer di dividere 100 per 6, la risposta sarà '16.6666667'. Compaiono ancora nove cifre e il risultato è arrotondato, come prima.

Priorità

In una espressione del tipo '2 + 4 — 5' gli operatori sono letti da sinistra a destra, come se dicessimo: 'sommare 4 a 6 e sottrarre 5, con risultato 1'. Ora provate questo:

```
PRINT 2 * 4 + 6
```

È come se diceste: 'Due per quattro fa otto che sommato a sei dà quattordici'. Il computer dà infatti come risultato 14. Ora invece introducete:

```
PRINT 2 + 4 * 6
```

La risposta fornita è 26. Evidentemente il calcolatore non legge gli operatori da sinistra verso destra. La macchina infatti esegue: 'Due sommato a quattro per sei è la stessa cosa che ventiquattro più due'. Prima esegue la moltiplicazione, poi l'addizione. Ecco altri due esempi:

```
PRINT 8 — 3 * 2
PRINT 8 * 3 — 2
```

si vede che l'operatore '*' è prioritario rispetto a '—'. I numeri impiegati sono piccoli. Ciò non per rendere più semplice i conti al VIC, ma a voi, in modo che possiate verificare le sue risposte!

Il computer obbedisce ad una regola di priorità. Essa consiste nel fatto che gli operatori '*' e '/' sono letti per primi, seguiti da '+' e da '—'. Prima di chiedere alla macchina il risultato dei seguenti esempi, usate la regola di priorità per capire quali saranno le risposte:

```
PRINT 8 + 12 / 3
PRINT 8 / 4 + 3
PRINT 8 * 4 + 3
```

A volte volete conoscere il risultato di una espressione del tipo:

$$\frac{2 \times 8}{4}$$

Questa espressione è scritta nel simbolo usuale matematico con un '×' che sta per 'moltiplica'. Nel computer si deve introdurre così:

```
PRINT 2 * 8 / 4
```

La seguente espressione necessita di un pò di attenzione:

$$\frac{12}{2 \times 3}$$

Dovete dividere dodici per il prodotto di due per tre. Potete semplificare il tutto scrivendo 12/6 con risultato 2. Ma se battete:

```
PRINT 12 / 2 * 3
```

otterrete 18, che è dodici diviso per due (ottenete 6), il risultato moltiplicato per 3 (= 18). Ciò non è quello che volete. Il modo corretto per scrivere quell'espressione è:

```
PRINT 12 / 2 / 3
```

Essa dà il risultato corretto '2', che è dodici diviso per due (= 6) e poi il tutto diviso ancora per tre. Anche se usate '×' nell'espressione originale, dovete introdurre '/' nella linea dell'istruzione.

Altri operatori

Vi è un quinto operatore aritmetico, il cui simbolo è '↑'. Battete questa linea e cercate di scoprire qual è la sua funzione:

```
PRINT 3↑2
```

Il risultato è '9', che è 3 per 3. In altre parole, due tre moltiplicati assieme. Il modo matematico di scrivere ciò è 3^2, che si legge '3 elevato alla seconda'. Potete anche chiamarlo 'tre al quadrato'. Ecco altri esempi:

```
PRINT 3↑4
PRINT 5↑2
PRINT 2*3↑2
```

Nell'ultimo troverete come risultato 18, che è 9 per due. Il computer legge prima '↑', ottenendo 9, poi moltiplica il risultato per due. L'operatore '↑' (o elevamento a potenza) ha la priorità su '*' e '/'.

L'esponente in questo operatore non deve essere necessariamente un intero:

```
PRINT 2 ↑ 2.5
```

Fornisce 5.65685425 come valore di $2^{2.5}$. Non siete in grado di verificare il risultato a mente, ma potete vedere che è compreso tra 2^2 (=4) e 2^3 (=8). Vi ricorderete che elevare un numero alla potenza di 0.5 significa estrarne la radice. Ecco due esempi:

```
PRINT 9↑.5
```

Fornisce la radice quadrata di 9, cioè 3. Per trovare la radice cubica di 8, scrivete:

```
PRINT 8↑ .333333333
```

Il risultato è 2, come ci si aspettava. Il fatto che la frazione un terzo presenta un numero infinito di cifre può costituire una difficoltà; dopo che avete battuto il nono 3, ne dovreste introdurre molti altri di più. Tuttavia nove cifre decimali approssimano abbastanza bene il valore di 1/3 e danno un risultato corretto. Ma se siete pigri e battete solo:

```
PRINT 8↑ .333
```

la risposta mostra che il valore .333 non è abbastanza vicino a 1/3. Provate a vedere cosa succede. C'è un modo per aggirare il problema delle frazioni:

```
PRINT 8↑ (1/3)
```

Questo ha lo stesso effetto dei nove tre, ma occorre scrivere molto meno. Potete rendervi conto di una nuova regola: il VIC inizia un qualsiasi calcolo prima di tutto trovando il valore delle espressioni contenute tra parentesi. Prima trova il valore di 1/3, ottenendo 0.333333333, e poi lo usa per calcolare 8↑ .333333333.

La regola delle parentesi si applica anche in altri casi. Confrontate i differenti risultati che si ottengono in:

```
PRINT 8 + 2 * 4 + 3
PRINT (8 + 2) * (4 + 3)
```

Nella prima espressione la moltiplicazione ha la priorità. Nella seconda invece si eseguono prima le addizioni perchè racchiuse tra parentesi. La moltiplicazione viene fatta dopo. Ciò mostra come sia possibile alterare l'ordine di esecuzione di operatori mediante l'uso di parentesi. Vi è un sommario di tutte le regole alla fine di questo capitolo.

Numeri piccoli e grandi

Questa sezione è per chi ama la matematica. Se volete saltarla, passate oltre alla sezione sulle Variabili. Qual è il numero più grande che un computer può elaborare? Vediamo. Potete rendervi conto di ciò mediante l'istruzione PRINT

```
PRINT 1000000
PRINT 10000000
PRINT 100000000
```

Fino a questo punto non ci sono problemi - e non dovete sorprendervi perchè è già stato detto che il VIC può usare numeri fino a nove cifre. Che ne dite di dieci?:

```
PRINT 1000000000
```

Potreste pensare che sia un trucco scrivere i numeri in un'altra forma, ma ciò è proprio quello che accade. È comparsa sullo schermo l'espressione '1E + 09'. Questo è il modo del computer per dire:

$$1 \times 10^9$$

Si dice che un'espressione di questo tipo è in una forma standard o esponenziale. È un modo per evitare di scrivere numeri elevati con lunghe righe di cifre. Ciò aiuta nella vostra indagine circa il numero massimo che il computer può prevedere. Provate a vedere come la macchina accetta questi esempi:

```
PRINT 1E + 10
PRINT 1E + 20
PRINT 1E + 30
PRINT 1E + 40
```

Solamente l'ultimo si rivela essere un numero troppo grande. Infatti dopo di esso compare il messaggio '? OVERFLOW ERROR'. Potete restringere la ricerca così:

```
PRINT 1E + 31
PRINT 1E + 32
```

e così via. Il numero più alto è tra 1E + 38 e 1E + 39. 1E + 38, cioè 1×10^{38}, è un 1 seguito da 38 zeri, o cento milioni di milioni di milioni di milioni di milioni di milioni. Potete restringere ulteriormente la ricerca provando:

```
PRINT 1.1E + 38
PRINT 1.2E + 38
```

e così via. Avendo stabilito il posto della prima cifra decimale, potete con-

tinuare con le altre. Continuate con la terza fino a dove volete.

Per tutte le applicazioni pratiche il massimo numero positivo è 1E+38, e il più piccolo negativo —1E + 38. Potete trovare allo stesso modo il più piccolo numero (cioè il più vicino allo zero che non è così piccolo da essere considerato zero). In questo caso usate il segno meno nella forma esponenziale. Se introducete:

```
PRINT 1E —1
```

sul video non appare il numero nella forma standard, ma in quella ordinaria, 0.1. L'espressione 1E — 1 significa 1×10^{-1}, e si può scrivere anche:

$$\frac{1}{10^1}, \text{ che corrisponde a 0.1 in decimale.}$$

Allo stesso modo, 1E — 2 significa 1×10^{-2}, che può essere scritto anche:

$$\frac{1}{10^2} = \frac{1}{100}, \text{ che corrisponde a 0.01 in decimale}$$

il computer risponde a 'PRINT 1E — 2' con '0.01' ma da qui in poi usa la forma esponenziale. Quando battete:

```
PRINT 1E — 3
```

compare '1E — 03'. Qual è il numero più piccolo che potete ottenere?

Variabili

I numeri finora trattati vengono immediatamente dimenticati dalla macchina. Se vogliamo che un numero sia ricordato per essere usato in un'altra occasione, dovete dire al VIC: 'Ecco il numero che voglio che tu memorizzi. Per aiutarti a fare ciò gli attribuirò un nome. Il numero è quattro e il suo nome è JIM. Ogni volta che mi riferisco a JIM, tieni presente che significa 4.

Un computer di fatto non comprende un simile linguaggio (anche se forse in futuro sarà possibile), pertanto dovete introdurre effettivamente:

```
JIM = 4
```

Il computer ha eseguito l'istruzione? Potete scoprirlo battendo:

```
PRINT JIM
```

La risposta perviene immediatamente ed è stampato 4. Chiedete di nuovo di stampare JIM per verificare che il suo valore non è stato perduto. Ciò accade solamente se viene meno l'alimentazione.

Avendo assegnato a JIM un valore numerico, la macchina lo considera esattamente come un numero. Provate alcuni di questi esempi:

```
PRINT JIM + 5
PRINT JIM + JIM
PRINT 2 * JIM
PRINT JIM ↑ 2
```

Elevare al quadrato JIM può sembrare strano, ma si può fare!

Possiamo chiedere al computer di ricordare due, tre o più numeri allo stesso tempo, avendo assegnato a ciascuno un suo nome. Consideriamone un altro, ad esempio:

```
KEN = 5
```

Poi scrivete:

```
PRINT JIM + KEN
PRINT 2 * KEN — JIM
```

ed altre espressioni simili.

Poichè la RAM può essere cambiata, potete chiedere al computer di modificare un qualsiasi numero in un altro. Avete bosigno semplicemente di una nuova istruzione:

```
JIM = 99
```

e da questo momento in poi il valore precedente di JIM, 4, è cancellato e quello nuovo diviene 99. Ora otterrete una serie di risultati diversi se introducete:

```
PRINT JIM + KEN
PRINT 2 * KEN — JIM
```

Scoprite ora cosa accade se scrivete:

```
JIM = KEN
```

Le parole che, come JIM e KEN, vengono usate per assegnare un nome ad un numero che si vuole far ricordare dalla macchina sono dette variabili. Questo termine è molto adatto perchè, come abbiamo visto, il loro valore può essere cambiato. Il VIC può memorizzare più variabili, ma non è necessario dare loro i nomi dei vostri amici. Per esempio, potete scrivere i nomi e i prezzi degli accessori del VIC:

```
JOYSTICK = 26000
PENNA LUMINOSA = 50000
MEMORIA RAM = 40000
NASTRO = 10000
```

Se volete calcolare il costo totale, battete:

```
PRINT JOYSTICK + PENNA LUMINOSA +
    MEMORIA RAM + NASTRO
```

o qualsiasi altra combinazione di oggetti che abbiate comprato, ottenendo il costo complessivo.

L'uso di nomi come variabili vi aiuta a rendere il programma più semplice e facile da capire, ma sfortunatamente occupa molto più spazio in memoria. Questo è un punto da chiarire. Il computer non ricorda tutto il nome, ma solo le prime due lettere. Così se dite al computer:

```
JIM = 5
```

e poi battete:

```
PRINT JI
PRINT JIG
PRINT JITTER
```

risponderà sempre con 5 in tutti i casi. Anche se una variabile è definita al massimo da due lettere, può averne anche una sola se lo desiderate. Il primo carattere di una variabile deve essere una lettera, mentre gli altri possono essere numeri. Queste sono le variabili accettate che iniziano con J:

```
J
JA, JB, JC,... JZ
J1,  J2, J3,... J0
```

Naturalmente questo vale per tutte le 26 lettere dell'alfabeto. Se volete usare JIM e KEN, potete farlo, ma la macchina li accetterà solo come JI e KE.

Alcune combinazioni di lettere non possono essere impiegate come variabili. TO, ON, OR, GO e IF sono parole del BASIC. Se provate ad usarle, otterrete dei messaggi d'errore. Vi sono anche alcune variabili speciali usate dal computer (vedere più avanti in questo capitolo). Anche parole più lunghe che hanno le prime due lettere costituite da espressioni di BASIC come ad esempio torta e oracolo, o che iniziano con espressioni BASIC come foresta e poker non sono ammesse.

Le variabili possono essere usate in diversi modi in un programma. Lo abbiamo già visto nel Tappeto Magico (Listato 3.3). Allora abbiamo impiegato delle variabili di un'unica lettera:

C per il colore del disco (linea 20)
X per la posizione della riga di dischi (linea 30)
Y per la posizione delle colonne dei dischi (linea 100)
J per contare quanti dischi sono stati stampati su di una riga (dalla linea 50 alla 80) o su di una colonna (dalla linea 110 alla 140). J è stata usata per dire al computer dove porre il Codice dei Caratteri e del Colore nella RAM.

Benchè le variabili in quel programma avessero tutti valori interi (cioè 0,22, 7777, 38497), abbiamo visto che sono ammessi anche numeri decimali (per esempio 2.45, 12.54), negativi e in forma esponenziale (1.23E+06).

Variabili intere

Le variabili fin qui descritte sono dette numeriche (a volte sono anche chiamate variabili a virgola mobile). Ve ne è un altro tipo, cioè le variabili intere. Queste non possono assumere valori decimali. Definiamo una variabile come intera ponendole come suffisso '%':

```
JIM% = 6
```

Ogni volta che assegnerete a JIM% un numero decimale, il computer ignorerà tutte le cifre dopo il punto decimale. Provate questo esempio:

```
JIM% = 7.89
PRINT JIM%
```

Se avete ancora in memoria JIM, chiedete di stampare JIM e JIM%. Vi accorgerete che JIM e JIM% sono variabili distinte con diversi valori. Usan-

do le variabili intere abbiamo a disposizione un nuovo gruppo di nomi, nel caso doveste usare più variabili contemporeamente.

Avreste potuto usare variabili intere anche nel programma del Tappeto Magico, ma vi è un inconveniente.

Il vantaggio principale delle variabili intere è che necessitano di uno spazio di memoria più ridotto. Se avete molti numeri e poca memoria può essere utile usare variabili intere. In questo modo velocizzate anche il programma perchè il computer impiega meno tempo nel verificare il valore della variabile (ammesso che non abbiate bisogno di cifre decimali). Generalmente può risultare noioso battere '%' dopo ogni nome, quindi preferiamo usare variabili numeriche ordinarie.

C'è un altro svantaggio nell'impiego delle variabili intere: esse non possono superare +32767 o essere minori di —32768. Questo intervallo è piuttosto grande per molti scopi, ma non per tutti. Potete usare variabili intere in tutti i modi, anche mischiandole in espressioni con variabili ordinarie.

Introduzione di valori

In questo capitolo avete imparato ad introdurre variabili e il loro valore premendo RETURN in modo che il computer li elabori direttamente. Come avete visto nel programma del Tappeto Magico, le variabili sono di uso frequente nei programmi. Nel Listato 4.1 è mostrato un programma che vi consente di cambiare il colore di un'immagine stampata.

```
10 PRINT"♥"
20 C = 2
30 PRINT"♥"
40 POKE 36879,220
50 FOR J = 7680 TO 8205 STEP 5
60 POKE J,88
70 POKE J + 30720,C
80 NEXT J
90 FOR K = 1 TO 2000:NEXT K
100 POKE 36879,27
110 GOTO 10
```

Listato 4.1 - Come cambiare i colori nel disegno.

Lanciate il programma e vedrete che l'immagine è in rosso. Ora cambiate il valore nella linea 20 per mutare il colore in giallo. La Tabella 7 (Appendice A) elenca i codici che vi servono.

Lo svantaggio di questo programma è che dovete modificare una linea per avere un differente colore. Sarebbe meglio poter effettuare la stessa operazione semplicemente introducendo un numero da tastiera. Battete una nuova linea 20:

```
20 INPUT C
```

Ora lanciate il programma. Non appare la figura, ma un punto interrogativo. Il VIC sta aspettando che voi introducete un numero. Per il momento il programma è bloccato. Battete un numero tra 0 e 7, secondo il colore desiderato e premete RETURN. Il numero che avete appena introdotto è assegnato alla variabile C. Il programma continua usando questo valore per C e la figura appare sullo schermo nel colore voluto. Il comando INPUT è veramente molto utile. Possiamo migliorarlo facendo apparire il messaggio:

```
20 INPUT ‘Qual è il Codice del Colore?’;C
```

Notate il punto e virgola. Ora chi usa il programma può sapere ciò che deve essere introdotto. Analizzeremo l'istruzione INPUT più avanti.

Variabili di stringhe

Vi è un altro tipo di variabili a vostra disposizione. Esse presentano il simbolo di ‘$’ alla fine del nome. Una variabile di stringa è esattamente un gruppo di caratteri - lettere, numeri, simboli grafici, di punteggiatura e così via. Per esempio:

```
NOME$= ‘MARCO ROSSI’
DATA$ = ‘29 FEBBRAIO 1985’
```

Introducete nel computer queste variabili di stringa battendo esattamente quanto scritto sopra. Ora date il comando:

```
PRINT NOME$; DATA$
```

Il computer visualizza il nome e cognome di Marco ed il suo compleanno. Notate che abbiamo detto alla macchina di stampare due stringhe, una dietro l'altra. I termini della lista di variabili sono separati da un punto e virgola. Sfortunatamente l'effetto ottenuto è rovinato dal fatto che il computer non pone uno spazio tra le due stringhe quando le visualizza. Se invece volete dividerle con uno spazio, potete porne uno alla fine della prima (NOME$= ‘MARCO ROSSI’) o aggiungerlo nella lista di variabili che devono essere stampate:

```
PRINT NOME$;‘ ’;DATA$
```

Quando volete introdurre una variabile stringa specificandone il contenuto, dovete porla tra virgolette ('). Una volta che una stringa è stata definita, potete usare solamente il suo nome.

Naturalmente non tutte le operazioni che si possono fare con le variabili numeriche o intere sono altrettanto eseguibili su variabili di stringa. Non potete, ad esempio, usare espressioni del tipo JIM$=3 * JOE$+JAN$ ↑ JON$. L'unico operatore che può agire sulle stringhe è il '+'. Se avete ancora NOME$ e DATA$ in memoria, introducete questa linea:

```
J$ = NOME$ + ' ' + DATA$
```

E poi:

```
PRINT J$
```

Il vostro comando ha detto alla macchina di aggiungere a NOME$ lo spazio e la DATA$, ponendo il risultato in J$. Sommare due stringhe significa semplicemente unirle assieme, nell'ordine in cui sono state elencate.

È importante notare che una stringa del tipo '123' che contiene il numero 123 non è uguale alla variabile intera o numerica 123. Potete scrivere:

```
PRINT 2 * 123
```

e otterrete come risposta 246, ma non potete fare la stessa cosa con una stringa. Per illustrare la differenza tra una variabile di stringa e gli altri tipi di variabili, provate questo esempio:

```
A$ = '123'
B$ = '456'
PRINT A$ + B$
```

Il risultato non è la somma 579, ma '123456'. Ci sono molte altre cose che potete fare con le stringhe e saranno spiegate nel capitolo otto.

Variabili speciali.

Vi sono alcuni nomi di variabili riservate eclusivamnte al computer. Esse sono ST, TI e TI$. Non parleremo di ST, mentre TI e TI$ sono estremamente utili. Tra i diversi circuiti che compongono la macchina ve ne è uno che genera sessanta pulsazioni al secondo il cui nome è orologio. Le pulsazioni sono contate e il loro numero è memorizzato nella variabile TI.

Come potete immaginare, TI è una forma abbreviata per TIME (= tempo). Usate sia TI o TIME per leggere questa variabile:

PRINT TI

Il risultato che ottenete dipende da quanto tempo avete acceso il computer. Il valore è il numero di sessantesimi di secondo che sono trascorsi da quando la macchina è stata accesa. Oppure sotto un altro punto di vista, TI diviso per sessanta dà il numero di secondi.

TI$ è una stringa lunga sei caratteri. I primi due a destra danno il numero di secondi da quando il calcolatore è stato acceso, le altre due i minuti, e le ultime due a sinistra le ore. Così se battete:

PRINT TI$

e il risultato è '013523' significa che il computer è rimasto in funzione per un'ora, 35 minuti e 23 secondi.

Anche se TI e TI$ partono da 0 quando il calcolatore è acceso, è possibile porle di nuovo a zero in qualsiasi istante. TI non può essere inizializzato direttamente, ma mediante TI$:

TI$ = '000000'

Tutti i sei caratteri sono eguagliati a zero. In questo modo TI e TI$ possono essere usati in giochi e in altri programmi, come vedremo più avanti. Se lo volete potete porre in TI$ un qualsiasi altro valore. Per esempio, 'TI$ = '123456' significa 12 ore, 34 minuti e 56 secondi. Questo è utile se avete un programma che necessita dell'ora . TI è posta a zero solo se si cambia TI$ in zero. Se TI$ è modificata in qualche altro valore, TI non cambia.

Il Listato 4.2 mostra un breve programma in cui è usata la variabile TI. Serve per stimare il tempo di reazione.

```
10 PRINT"J"
20 POKE 36879,122
30 FOR J =1TO INT(RND(1)* 10000):NEXTJ
40 POKE 7910,127
50 POKE 38630,2
60 TI$ = "000000"
70 GET A$
80 IF A$ = "" THEN 70
90 PRINT"J"
100 POKE 36879,47
110 PRINT"■IL VOSTRO TEMPO DI REAZIONE E':"
115 PRINT:PRINT TI/60;"SECONDI"
120 GOTO 120
```

Listato 4.2 - Valutiamo il tempo di reazione.

Dopo aver scritto PRINT' nella linea 110, premete CTRL assiema a 2. Questo vi darà una 'E' in reverse sullo schermo che il computer interpreta come 'stampa le lettere in bianco'. Lanciate il programma. Lo schermo è cancellato alla linea 10. La 20 lo pone in rosso con i bordi gialli. Vi è poi un'interruzione per un periodo di tempo casuale dopo il quale appare un carattere al centro dello schermo (linee 40 e 50). Il programma pone a zero TI (cioè azzera TI$) non appena compare il carattere. Quando lo vedete, premete un tasto qualsiasi. Non appena premuto il tasto, il programma pone lo schermo e il bordo in reverse e poi visualizza il vostro tempo di reazione, usando TI. Il valore di TI è diviso per 60, dando il tempo di reazione in secondi.

Un aspetto interessante del programma è l'uso del comando GET. Come INPUT, esso consente all'utente di interagire con il programma mentre è in esecuzione. Vi sono tre differenze fondamentali tra INPUT e GET:

(1) GET non attende. Quando il computer raggiunge la linea in cui è presente GET, la tastiera è pronta. La variabile che deve essere introdotta (per esempio A se il comando è GET A o A$ se è GET A$) assume il valore di qualsiasi carattere battuto in quell'istante. Se non battete alcun tasto la variabile assume il valore nullo. Se la variabile è numerica (ad es. A) diventa uguale a zero, se è una variabile di stringa, diventa una stringa vuota ("").
(2) GET funziona senza premere RETURN. È una conseguenza di quanto già detto prima. Al contrario, INPUT attende l'introduzione del dato quanto volete; premete RETURN quando avete finito.
(3) GET può accettare un solo carattere. Ciò segue dal punto precedente, perchè il computer può riconoscere un solo carattere per volta.

Nel programma del tempo di reazione usate A$. Questo permette all'utente di battere qualsiasi tasto. Se aveste usato GET A, avreste dovuto introdurre un numero - un altro catrattere avrebbe causato un messaggio di errore! Quando il programma raggiunge GET A$ e non viene premuto alcun tasto, A$ diventa una stringa nulla. La linea 80 verifica se A$ è nulla oppure no. Se lo è (IF A$=" ") il programma ritorna a leggere il carattere introdotto da tastiera. In caso contrario, (cioè è stato introdotto un carattere) il programma continua alla linea successiva valutando la vostra reazione.

Un altro aspetto interessante del programma sul tempo di reazione è la diramazione alla linea 80. Se viene premuto un tasto il programma continua con il conteggio, altrimenti ritorna indietro all'introduzione da tastiera. Queste due possibilità sono il risultato dell'uso della seguente sequenza di parole:

IF (condizione) THEN (azione)

La condizione è A$=" ". Se la condizione è vera, (cioè se A$ è una stringa nulla), il computer esegue l'azione specificata. In questo caso è 'GOTO 70'. Se A$ è nulla, ritorna indietro alla linea 70 e verifica il carattere introdotto. Continua a fare questo fino a che la condizione non si verifica. Se la condizione è vera, (cioè A$ non è una stringa nulla), il calcolatore non esegue l'azione specificata, ma passa semplicemente alla linea successiva. 'IF...THEN' è una sequenza di parole che useremo molto in questo libro.

Riassunto

In questo capitolo vi è stato spiegato come:

— fare eseguire al VIC tutti i tipi di calcoli
— elaborare numeri molto grandi e molto piccoli
— usare i nomi di variabili
— usare variabili numeriche, intere, e di stringa
— misurare il tempo, mediante TI e TI$
— usare i comandi INPUT e GET
— scrivere un programma che contiene un blocco decisionale, usando IF (condizione) THEN (azione)

Avete anche imparato:

— che gli operatori aritmetici sono +, —, *, /, e ↑
— che la regola di priorità nei calcoli è:

Prima le espressioni nelle parentesi
Poi ↑
Quindi, * e / (da sinistra a destra)
Per ultimo, + e —.

Capitolo 5
Suoni fantastici

Nel capitolo tre abbiamo visto che il Chip di Interfaccia Video (Video Interface Chip, dal quale prende nome il VIC) vi consente di ottenere i migliori effetti visivi con un minimo di programmazione. Troverete ulteriori informazioni circa questo aspetto del Chip nel capitolo sette, mentre ora esamineremo un'altra caratteristica della sua natura - i generatori di suoni.

Il VIC possiede tre generatori di note e uno di rumore, che possono essere controllati indipendentemente uno dall'altro. Essi producono segnali che alimentano i circuiti audio del televisore. Se avete abbassato il volume per evitare il sibilo che è emesso quando si accende per la prima volta, ora alzatelo di nuovo, pronto per il primo programma dimostrativo mostrato nel Listato 5.1.

```
10 S = 36875
20 V = 36878
30 INPUT"FREQUENZA,VOLUME";P,Q
40 POKE S,P
50 POKE V,Q
60 FOR K = 1 TO 500
70 NEXT K
80 POKE S,0
90 POKE V,0
100 GOTO 30
```

Listato 5.1 - Programma dimostrativo per l'uso dei generatori di suoni.

Prima di lanciare questo programma, analizzatelo per vedere cosa realizza e come funziona. Nelle linee 10 e 20 si inizializzano due variabili, S e V. Esse sono gli indirizzi di due parti del chip del suono. S è l'indirizzo di uno dei tre generatori di note (Tabella 8 dell'Appendice A). V è l'indirizzo del registro di controllo del volume. Questo varia il volume del suono proveniente da tutti e quattro i generatori. Non è possibile variarli indipendentemente. Come mostra la Tabella 8, la frequenza del suono proveniente dai generatori dipende dal valore posto nei rispettivi indirizzi mediante una istruzione POKE. Questo valore deve essere compreso tra 128

e 254. Più il valore è alto, maggiore è la frequenza. Porre un valore tra 0 e 127 significa lasciare spento il generatore, mentre 255 fornisce lo stesso effetto di 128. Il controllo del volume prevede un valore tra 0 e 15. Minore è il numero, più basso è il volume, quindi 0 corrisponde al silenzio. Porre mediante l'istruzione POKE 0 nel controllo del volume non è la stessa cosa che porre 0 nel generatore di suoni. Porre il volume a 0 non spegne i generatori. Se lo si aumenta ulteriormente, i generatori emettono suoni esattamente come prima. A proposito, se il suono diventa insopportabile, vi è un modo semplice per interromperlo! Spegnete tutti i generatori e riducete il volume a 0 premendo il tasto RUN/STOP con RESTORE.

In questo programma sono usate le variabili S e V, in modo che potete precisare quali indirizzi introdurre alle linee 40 e 50. Il programma con la linea 30 vi consente di introdurre due numeri, uno per la frequenza (da 128 a 254), l'altro per il volume (da 0 a 15). Notate che la linea è composta da:

La parola INPUT,

seguita da un messaggio compreso tra virgolette (a volte è chiamato 'suggerimento' perché suggerisce all'utente cosa introdurre),

seguita da un punto e virgola (;), che è omesso se non c'è il messaggio,

seguita da una lista di una o più variabili, separate da virgole; le variabili che compaiono possono essere mischiate, includendo variabili numeriche, intere o di stringa in qualsiasi ordine.

Quando arriva a questa linea del programma, il computer stampa il messaggio di suggerimento, aspettando l'introduzione della frequenza e il volume. Battete due numeri (secondo i valori previsti) separati da una virgola. Poi premete RETURN.

Se vi dimenticate e premete RETURN dopo il primo numero, il calcolatore stampa '??', indicando che sta ancora aspettando il secondo valore. Introducetelo e premete di nuovo RETURN. Poichè questa istruzione di INPUT ha due variabili numeriche al suo interno, dovete assicurarvi di battere due numeri, non lettere. Se per errore introducete delle lettere, il computer vi dà una seconda possibilità visualizzando 'REDO FROM START'e ripetendo il messaggio e il punto interrogativo.

Le linee 40 e 50 memorizzano mediante POKE i valori che avete appena scelto, e il suono inizia. Le linee 60 e 70 fanno contare il computer da 1 a 500. Questo è un loop di ritardo. È un modo per ritardare il calcolatore per qualche attimo ed impedirgli di spegnere immediatamente il suono alla linea 80. Vi consente di ascoltarlo per circa un secondo.

È importante spegnerlo alla fine del programma. Questo avviene alle righe 80 e 90. Dopo di ciò la macchina ritorna alla linea 30 per chiedervi

i dati circa il suono successivo da eseguire. La Tabella 9 dell'Appendice A mostra quali valori dare per ottenere particolari note della scala musicale. Il programma usa il generatore di suoni Tenore, pertanto consultate la colonna centrale dei caratteri. I valori che compaiono nella tabella sono gli interi più vicini a quelli reali, perchè i generatori possono accettare solo interi. Per questa ragione alcune note possono avere una frequenza leggermente diversa da quella esatta, ma la differenza di solito non è percepita se non da musicisti esperti.

La Tabella 9 mostra anche i valori usati dagli altri generatori di suoni. Per provarne gli effetti, cambiate la linea 10 come segue:

```
10 S = 36876
```

per il generatore Soprano o in:

```
10 S = 36874
```

per il Basso.

Il quarto generatore di suoni è diverso dagli altri tre. Invece di una nota, produce un rumore. Tecnicamente si dice 'rumore bianco'. L'uscita di questo generatore è filtrata in modo che il suono vari, dal ruggito di un torrente di montagna (P = 128) al sibilo di un vapore che esplode per una forte compressione (P = 254). Modificate la linea 10 del programma in:

```
10 S = 36877
```

e ponete in funzione il generatore di rumori.

Variazione di frequenza e volume

I suoni o i rumori di solito non consistono semplicemente in note o rumori improvvisi e a frequenza e volume costante. Frequenza e rumore possono diminuire o aumentare. Provate il programma del Listato 5.2.

```
10 S = 36875
20 V = 36878
30 POKE V,15
40 FOR J = 128 TO 254
50 POKE S,J
60 FOR K = 1 TO 10: NEXT K
70 NEXT J
80 POKE S,0:POKE V,0
```

Listato 5.2 - Come aumentare o diminuire la frequenza.

Qui con un loop introduciamo nel generatore di suoni mediante l'istruzione POKE valori diversi. Ogni volta che è incrementata J, a partire da 128 fino a 254, la frequenza aumenta. Vi è un brevissimo ritardo nell'esecuzione del loop esterno. Esso è causato dal loop di ritardo della linea 60. Questo è un buon esempio di loop nidificati. Il ciclo esterno (FOR I = 128 TO 254 ... NEXT J) include un ciclo interno (FOR K = 1 TO 10 ... NEXT K). Quando due loop sono nidificati, la prima variabile usata come contatore (J, in questo esempio, alla linea 40) è anche l'ultima ad essere usata (NEXT J, alla 70).

Potete modificare la linea 40 in modo da ottenere un suono di frequenza decrescente:

```
40 FOR J = 254 TO 128 STEP —1
```

Il loop questa volta sta contando alla rovescia. Per dire al computer come contare, si usa la parola addizionale 'STEP' (= passo). Contare all'indietro di 1 è indicato da 'STEP —1'. Questa parola può essere usata anche con altri valori. Se volete che il suono si alzi velocemente, potete cambiare la linea 40 ad esempio così:

```
40 FOR J = 128 TO 254 STEP 10
```

J assume i valori 128, 138, 148, ... fino a 248. Il passo successivo a 248 sarebbe 258, ma visto che è maggiore del massimo limite specificato, il ciclo non si ripete con questo valore. Provate questo esempio nel programma per vedere come riesce. Provate a variare anche i valori iniziali e finali della linea 40, usando anche diversi decrementi mediante 'STEP'. In un 'FOR...NEXT' senza 'STEP' il passo è posto automaticamente uguale a +1.

Ora vediamo altri effetti che si possono ottenere con semplici modifiche al programma. Introducete una nuova linea:

```
75 GOTO 40
```

Questa fa ritornare il computer alla linea 40 e gli fa ripetere il loop. Se lo STEP nella linea 40 ha un valore di circa 25, otterrete un rumore simile a un potente motore in funzione. Ricordate di premere RUN/STOP se siete stanchi di ascoltarlo!

Il programma nel Listato 5.3 ha alcune caratteristiche in comune con il programma precedente (Listato 5.2).

Dalla linea 10 potete vedere che vengono usati i generatori del rumore e di Soprano (N = 36877). La prima parte produce una nota di frequen-

```
10 S = 36876: N = 36877
20 V = 36878
30 FOR J = 230 TO 180 STEP -1
40 POKE S,J
50 POKE V, (230-J)/4
60 FOR K = 1 TO 30:NEXT K
70 NEXT J
80 POKE S,0
90 POKE V,15
100 POKE N,180
110 FOR K = 1 TO 250: NEXT K
120 FOR J = 15 TO 1 STEP -1
130 POKE V,J
140 FOR K = 1 TO 200:NEXT K
150 NEXT J
160 POKE V,0
170 POKE N,0
180 END
```

Listato 5.3 - La caduta della bomba è il primo passo per costruire il programma del Raid Aereo.

za decrescente (vedere la linea 30). All'istruzione 50 c'è una serie di addizioni all'interno di un loop. La nota non solo è di frequenza decrescente, ma aumenta anche il volume. Questo parte da 0, quando J vale 230, perchè (230-J)/4 = 0/4 = 0. Notate l'uso delle parentesi che permettono al computer di eseguire la sottrazione (230-J) prima della divisione. Alla fine del loop, quando J = 180, il valore assegnato al controllo del volume mediante POKE è (230 = 180)/ = 50/4 = 12.

L'effetto sovrapposto di una successione di note discendenti di volume crescente simula assai bene il rumore di una bomba che cade, ed ecco perchè il programma è stato chiamato Raid Aereo. Nella linea 80 il generatore viene spento. È l'inizio dell'esplosione. Il volume è alzato per mezzo secondo (linea 110) e poi ridotto a zero, e l'eco dell'esplosione si attenua (120-150). Finalmente (linee 160 e 170) cala il silenzio non appena il volume è ridotto a zero e il generatore di rumore viene spento.

Il programma illustra l'idea di inviluppo del suono. Quello di una bomba che cade è semplice in quanto decresce in modo costante. Potete vederlo nella figura 5.1, che mostra la sequenza degli effetti. Diagrammi di questo tipo sono molto utili nella realizzazione di programmi che fanno uso di suoni. L'inviluppo dell'esplosione è più complicato, dal momento che il livello del suono è dapprima costante e poi decresce.

Se volete un raid aereo continuo, aggiungete al programma queste linee:

```
180 FOR K = 1 TO INT(RND(1) * 6000)
190 NEXT K
200 GOTO 30
```

Ciò vi fornisce un ritardo di lunghezza variabile, determinato da un numero generato casualmente. Dopo questo ritardo cade un'altra bomba.

Per completare l'illusione e renderla anche visiva, ecco alcune linee da

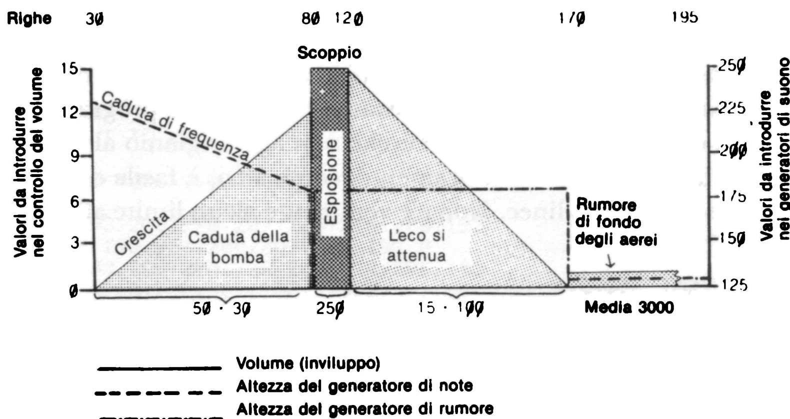


Fig. 5.1 - La carta del suono per il programma del Raid Aereo mostra la variazione del volume e della frequenza.

aggiungere al programma:

```
5 PRINT '[♥]': POKE 36879,8
```

Questa cancella lo schermo e lo pone in nero come il bordo (è un raid aereo notturno!)

```
135 POKE 36879,9
140 FOR K = 1 TO 100: NEXT K
145 POKE 36879,24
147 FOR K = 1 TO 100: NEXT K
```

In questo modo si spezza il ritardo originale della linea 140 in due parti. Le linee 135 e 145 alternativamente pongono lo schermo bianco con il bordo nero e poi nero con il bordo bianco.

```
165 POKE 36879,8
```

pone lo schermo e il bordo ancora neri. Per migliorare l'effetto spegnete le luci della stanza. Lanciate il programma. Noterete apparire sullo schermo luci spettacolari ogni volta che cade una bomba. Manca ancora qualche cosa? Certamente, il rumore dei bombardieri. Cambiate queste linee:

```
160 POKE V,1
170 POKE N, 128
```

e aggiungete:

```
195 POKE V,0: POKE N,0
```

Ora potete ascoltare il rombo dei bombardieri mischiato al sibilo delle bombe che cadono e alle esplosioni. Il Listato 5.4 mostra il programma del Raid Aereo con queste modifiche. Perchè non aggiungiamo alcune stelle al cielo notturno? Questo miglioramento, come altri, è facile da ottenere introducendo una o più linee. Non c'è veramente alcun limite alla versatilità del VIC.

```
5 PRINT"J":POKE 36879,8
10 S = 36876:N = 36877
20 V = 36878
30 FOR J = 230 TO 180 STEP -1
40 POKE S,J
50 POKE V, (230-J)/4
60 FOR K = 1 TO 30:NEXT K
70 NEXT J
80 POKE S,0
90 POKE V,15
100 POKE N,180
110 FOR K = 1 TO 250:NEXT K
120 FOR J = 15 TO 1 STEP -1
130 POKE V,J
135 POKE 36879,9
140 FOR K = 1 TO 100:NEXT K
145 POKE 36879,24
147 FOR K = 1 TO 100: NEXT K
150 NEXT J
160 POKE V,1
165 POKE 36879,8
170 POKE N,128
180 FOR K = 1 TO INT(RND(1)*6000)
190 NEXT K
195 POKE V,0:POKE N,0
200 GOTO 30
```

Listato 5.4 - Il programma completo del Raid Aereo.

Tutti i sistemi funzionano

Abbiamo già usato assieme un generatore di suono e uno di rumore. Si può migliorare la musica operando con tutti e tre i generatori di note contemporaneamente. Ma prima di fare ciò, nel programma della Giostra (Listato 5.7), occorre che sappiate in che modo il computer elabora i dati.

Se volete che siano eseguite più note in rapida successione e visto che non se ne possono introdurre più di tre alla volta, non è possibile che voi rimaniate alla tastiera per battere tutti i numeri, come avete fatto nel programma del Listato 5.1. I numeri (o i dati) devono essere già nel calcolatore, pronti per essere usati nel programma. Un modo per memorizzare dati nella macchina è l'istruzione DATA. Potete capire come usarla analizzando il programma del Listato 5.5.

Le linee 100 e 110 sono separate dal programma principale, come potete vedere dal fatto che nella 90 compare la parola 'END'. In esse compaio-

```
10 S = 36875: V = 36878
20 POKE V,15
30 FOR J = 1 TO 31
40 READ N
50 POKE S,N
60 FOR K = 1 TO 200:NEXT K
70 NEXT J
80 POKE S,0:POKE V,0
90 END
100 DATA 193,206,200,209,206,214,206,193
105 DATA 0,193,206,200,209,206,193,0
110 DATA 193,206,200,209,206,214,206,193
115 DATA 0,218,200,209,206,193,0
```

Listato 5.5 - Memorizzazione delle note di un motivo in una istruzione DATA.

no i valori che devono essere posti negli indirizzi del registro Tenore mediante il comando POKE. L'istruzione DATA contiene 31 numeri, uno per ogni nota del motivo. Le linee 30 e 70 sono un loop ripetuto 31 volte. Il comando READ N assegna alla variabile N uno dei valori dell'istruzione DATA. I valori cambiano ogni volta che inizia il loop.

La prima volta, ad esempio, N è pari a 193. Alla linea 50 è posto nel generatore di suoni Tenore mediante POKE. Viene generato dall'altoparlante il do centrale. La volta successiva, N assume il secondo valore della linea 100. Questo, 206, è introdotto nel generatore, fornendo la nota mi. Pertanto ogni valore viene letto di volta in volta. Quando c'è '0' il risultato è un periodo di silenzio. Il fatto che i dati sono su due linee separate non importa. L'istruzione READ continua alla linea successiva come se i dati fossero tutti sulla stessa riga. Il computer legge mediante READ ogni singola voce dei dati una per volta, suonando la nota corrispondente. Arrivato a 31 (compresa la pausa) il motivo è completo e il programma finisce.

Ora provate ad aggiungere questa linea:

```
85 GOTO 20
```

Quando lanciate il programma ascoltate il motivo una volta, ma alla fine appare sullo schermo '?OUT OF DATA ERROR IN 40'. Tutti i 31 dati sono stati letti durante la prima esecuzione della musica. Ora non vi sono più elementi da leggere. Fortunatamente c'è un modo per far ritornare indietro il computer all'inizio della lista:

```
83 RESTORE
```

Questo comando permette al calcolatore di ritornare a leggere da capo la lista, così che potete ascoltare il motivo, ripetuto quante volte volete. Anche se il computer non ha terminato di leggere mediante READ un gruppo di istruzioni DATA, RESTORE lo fa ricominciare dall'inizio comunque.

Vettori

Un altro modo per memorizzare i dati in un programma è porli in un vettore. Questo, come l'istruzione DATA, è una specie di lista. Un vettore contiene delle variabili che hanno tutte lo stesso nome. Così come potete avere una variabile chiamata JIM, lo stesso nome si può usare per un vettore. Una variabile JIM può avere un solo valore, mentre un vettore ne può assumere diversi contemporaneamente.

Un vettore si definisce 'dimensionandolo'. In altre parole si decide la sua lunghezza, cioè quante variabili può memorizzare. Per esempio, se volete che assuma dieci differenti valori, introducete la seguente linea:

```
10 DIM JIM(10)
```

Se avete ancora il programma precedente in memoria, liberatevene battendo NEW, poi scrivete la linea sopra indicata. Sembra che non accada nulla quando premete RETURN, ma il VIC ha riservato una parte della sua memoria a disposizione del vettore di dieci variabili. Il fatto che il nome JIM è seguito da un numero da solo o tra parentesi segnala al computer che JIM è un vettore, non una variabile numerica. Se volete, potete avere anche una variabile numerica chiamata JIM assieme al vettore, e la macchina li tratterà in modo separato. Le regole circa i nomi delle variabili si applicano anche ai vettori. JIM(10) è formato da variabili numeriche, JIM%(10) da interi, mentre JIM$(10) da stringhe. Non potete mischiare variabili di genere diverso nello stesso vettore. Possono esistere contemporaneamente tutti e tre i tipi di vettori, anche se usare lo stesso nome potrà confondere voi, e non il computer! Le altre regole circa i nomi permessi sono esposte nel capitolo quattro.

Potete considerare il vettore appena definito come una lista, come appare nella fig. 5.2. Ogni suo elemento ha un numero, il suo indice. Noterete che di fatto può contenere 11 elementi, non 10, perchè la lista inizia con l'indice zero. Generalmente non si utilizza il numero zero, e si procede come se la lista iniziasse realmente con l'indice 1. Ora per poter introdurre alcuni numeri nel vettore scrivete:

```
20 FOR J = 1 TO 10
30 INPUT JIM(J)
40 NEXT J
50 FOR J = 1 TO 10
60 PRINT JIM(J)
70 NEXT J
80 END
```

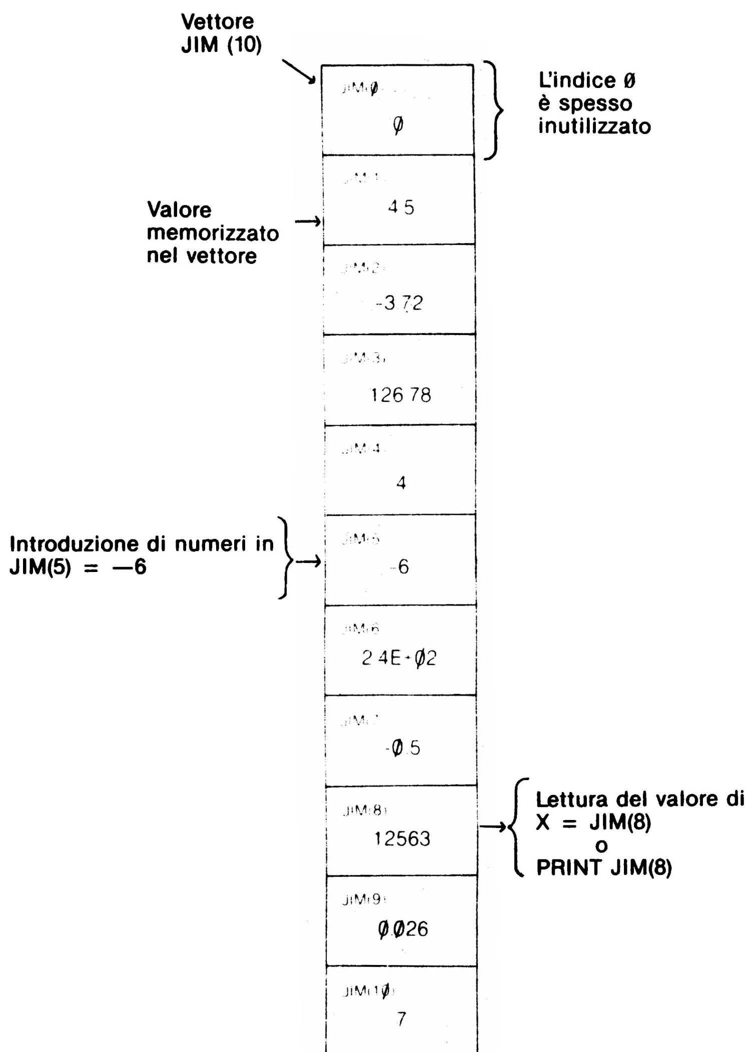


Fig. 5.2 - Un vettore ad una dimensione chiamato JIM. È un vettore numerico, quindi può contenere tutti i tipi di numeri.

Questo programma è diviso in due parti. La prima metà riceve i dati non appena li avete introdotti dalla tastiera e li pone nel vettore JIM (per evitere di usare l'espressione troppo lunga 'vettore JIM', da ora in poi diremo semplicemente 'JIM()').

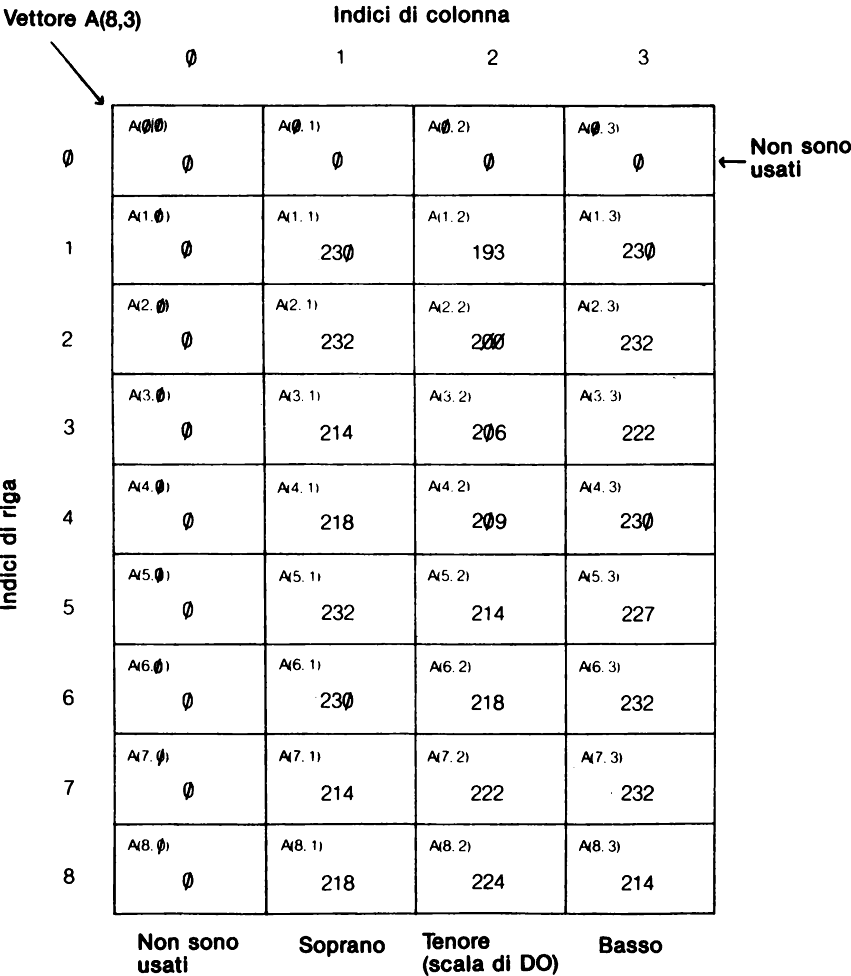
La seconda parte del programma prende i numeri da JIM() e li visualizza uno per uno sullo schermo. Lanciate ii programma. Dovete introdurre dalla tastiera 10 numeri, uno dopo l'altro e ognuno di essi viene me-

morizzato come in una lista, partendo da JIM(1) a JIM(10). Appena avete finito di scriverli, il programma continua e stampa tutti i numeri.

Ora modificate la seconda parte del programma come segue:

```
50 INPUT 'NUMERO INDICE';X
60 PRINT JIM(X)
70 GOTO 50
```

Lanciate il programma. Per prima cosa, introducete dieci numeri, come prima. La seconda parte vi permette di scoprire cosa è stato memorizzato



Vettore A(8,3) — Indici di colonna / Indici di riga

Indici di riga	Ø	1	2	3	
Ø	A(Ø.Ø) Ø	A(Ø. 1) Ø	A(Ø. 2) Ø	A(Ø. 3) Ø	← Non sono usati
1	A(1.Ø) Ø	A(1. 1) 23Ø	A(1. 2) 193	A(1. 3) 23Ø	
2	A(2.Ø) Ø	A(2. 1) 232	A(2. 2) 2ØØ	A(2. 3) 232	
3	A(3.Ø) Ø	A(3. 1) 214	A(3. 2) 2Ø6	A(3. 3) 222	
4	A(4.Ø) Ø	A(4. 1) 218	A(4. 2) 2Ø9	A(4. 3) 23Ø	
5	A(5.Ø) Ø	A(5. 1) 232	A(5. 2) 214	A(5. 3) 227	
6	A(6.Ø) Ø	A(6. 1) 23Ø	A(6. 2) 218	A(6. 3) 232	
7	A(7.Ø) Ø	A(7. 1) 214	A(7. 2) 222	A(7. 3) 232	
8	A(8.Ø) Ø	A(8. 1) 218	A(8. 2) 224	A(8. 3) 214	
	Non sono usati	Soprano	Tenore (scala di DO)	Basso	

Fig. 5.3 - Struttura di un vettore bidimensionale contenente i dati trasferiti da una istruzione DATA nel programma della Giostra.

in ciascun elemento di JIM(). Quando vi si chiede il numero dell'indice, battete un numero tra 1 e 10.

Quindi il computer stampa il numero memorizzato in JIM sotto quell'indice. Se battete '0' avete come risposta '0', perchè non è stata assegnato alcun numero a JIM(0). Se introducete '11' o qualche altro numero maggiore di 10, avrete come risposta 'BAD SUBSCRIPT ERROR'. Il vettore è stato dimensionato a dieci elementi e non di più.

Questo programma dimostra uno dei vantaggi dei vettori. Potete ottenere qualsiasi numero semplicemente fornendone l'indice. Per esempio, se volete il quarto numero del vettore, usate solo il termine JIM(4). Non c'è alcun bisogno che vi preoccupiate dell'intera lista, come nell'istruzione DATA.

Un altro modo per introdurre gli elementi in un vettore è quello di trasferirli da una istruzione DATA:

```
10 DIM A$(7)
20 FOR J = 1 TO 7
30 READ A$(J)
40 NEXT J
50 FOR J = 7 TO 1 STEP —1
60 PRINT A$(J);
70 NEXT J
80 END
100 DATA 'E', 'R', 'O', 'T', 'T', 'E', 'V'
```

Questo esempio usa un vettore stringa, A$(), dimensionato a dieci elementi (veramente non c'è alcun bisogno di dimensionare un vettore se ha meno di 11 indici, includendo lo zero, in modo che ogni vettore non dimensionato viene posto automaticamente a 10). Dal momento che A$() è un vettore stringa, l'istruzione DATA deve contenere delle stringhe, ed è questa la ragione delle virgolette. Le linee 20 e 40 introducono in A$() i dati della linea 100. Le linee 50 e 70 li stampano, ma in ordine inverso per formare la parola. Notate l'uso del punto e virgola (;) dopo 'PRINT A$(J)'. Ciò impedisce al computer di iniziare una nuova linea ogni volta che stampa la stringa successiva. Tutte vengono stampate sulla stessa linea, una subito dopo l'altra, così da formare la parola a partire dalle singole lettere.

Il vantaggio di trasferire gli elementi in un vettore a partire da una istruzione DATA è che sono più semplici da maneggiare. Potete accedere a ogni elemento usando il suo indice. Sono possibili anche altri tipi di operazioni che non si ottengono da una istruzione DATA, come la stampa in ordine inverso di questo programma.

Il prossimo, la Giostra, usa un vettore a due dimensioni. Potete pensar-

lo non come una lista, ma una tabella, con linee e colonne. Ci sono due gruppi di indici, uno per le righe e uno per le colonne. Il vettore A() usato nella Giostra ha 8 linee e 3 colonne (fig. 5.3) pertanto è dimensionato dall'istruzione 'DIM A(8,3). Abbandoniamo ora la trattazione dei vettori per scoprire le delizie della Giostra.

La Giostra

Questo è un programma che suona una musica tipica di una giostra. Fa uso di due vettori, N() e A(). N() contiene la nota che deve essere suonata, ed ha al massimo 12 numeri. Ognuno di essi corrisponde al valore delle otto note dal do centrale al successivo. A() contiene i numeri che devono essere posti nei generatori di suoni mediante l'istruzione POKE per produrre le note desiderate.

```
10 DIM N(12),A(8,3)
20 S1=36876:S2=36875:S3=36874
25 S4=36877:V=36878
30 FOR J = 1 TO 12
40 PRINT"NO.";J;:INPUTN(J)
50 NEXT J
60 FOR J = 1 TO 8
70 FOR K = 1 TO 3
80 READ A(J,K)
90 NEXT K
100 NEXT J
110 FOR J = 1 TO 12
120 POKE S2,A(N(J),2):POKES4,253
130 POKE V,15
140 GOSUB 1000
150 GOSUB 1000
160 POKE S1,A(N(J),1:POKE S3,A(N(J),3)
170 GOSUB 1000
180 POKE S1,0:POKE S3,0
190 GOSUB 1500
200 POKE S1,A(N(J),1):POKE S3,A(N(J),3)
210 GOSUB 1000
220 POKE S1,0:POKE S3,0
230 IF J/3=INT(J/3)AND F=0THEN F=1:GOTO140
240 F = 0
250 POKE S1,0:POKE S4,0
260 GOSUB 1500
270 NEXT J
280 GOTO 110
1000 FOR K=1 TO 100:NEXT K:RETURN
1500 FOR K=1 TO 30:NEXT K:RETURN
2000 DATA 230,193,230,232,200,232,214
2005 DATA 206,222,218,209,230
2010 DATA 232,214,227,230,218,232,214
2015 DATA 222,232,218,224,214
READY.
```

Listato 5.6 - La Giostra.

Il programma inizia con il dimensionamento dei vettori (linea 10). È necessario battere DIM una volta sola, seguito dalla lista dei vettori, se-

parati da virgole. N() è una lista con 12 indici (non considerando N(0)), mentre A() ha la struttura mostrata in fig. 5.3. La linea 20 definisce alcune variabili che sono gli indirizzi dei tre generatori di suoni (S1, S2, S3), del rumore (S4) e del controllo del volume (V).

Quando il programma è in esecuzione, viene richiesto all'utente di introdurre dalla tastiera il motivo (linee 30 e 50) che deve essere memorizzato in N(). Battete dodici numeri, tutti compresi tra 1 e 8. Ecco la corrispondenza tra numeri e note:

Numeri	1	2	3	4	5	6	7	8
Note	do	re	mi	fa	sol	la	si	do

Non preoccupatevi di introdurre un motivo particolare. La Giostra è in grado di trasformare in un'attraente melodia anche una scala ascendente.

Non appena sono stati battuti dodici numeri, il programma riempie A() con le frequenze indicate dall'istruzione DATA (linee 60 e 100). Ciò si ottiene usando loop nidificati. Quello più esterno incrementa J da 1 a 8, per esaurire ogni riga, una per volta. Per ognuna di esse, il loop interno incrementa K da 1 a 3 per riempire tutte le colonne. Pertanto il vettore è riempito come se aveste copiato tutti gli elementi dalle linee 2000 e 2010 riga per riga, andando da sinistra a destra e dall'alto al basso.

Ecco che inizia la musica! Vi è un lungo loop (dalla linea 110 alla 270) che viene ripetuto una volta per ciascuna nota. Prima di tutto entra in funzione il generatore Tenore che produce la nota specificata dalla colon-

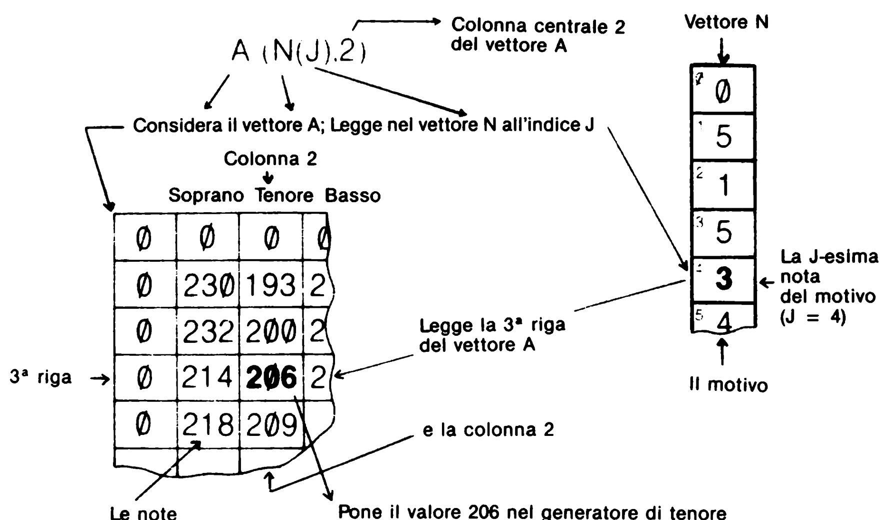


Fig. 5.4 - Interpretazione del computer dell'espressione A(N(J),2), dove J vale 4.

na centrale del vettore. La figura 5.4 mostra come il computer interpreta l'espressione A(N(J),2). Il volume è posto al massimo (linea 130), pertanto la Giostra suona il più alto possibile. Il generatore Tenore domina sempre gli altri due, mettendo in risalto il canto rispetto all'accompagnamento.

Il secondo comando della linea 120 fa in modo che il generatore di rumore produca l'effetto del motore che si sente quando ascoltate la musica di una giostra.

La nota è ottenuta usando una subroutine (linea 1000), che produce un loop di ritardo. Si accede a questa subroutine dal programma principale mediante l'istruzione 'GOSUB 1000' alla linea 140. Quando il ritardo è terminato, il comando 'RETURN' alla fine della linea 1000 fa ritornare indietro il computer alla linea successiva a quella alla quale si era interrotto il programma principale, cioè alla 150. Qui trova ancora 'GOSUB 1000', quindi produce un altro ritardo e ritorna alla 160. Abbiamo usato due volte la subroutine per ottenere un ritardo di lunghezza doppia.

Si ode ancora la nota di tenore, ed ora aggiungiamo gli altri due generatori di suono (linea 160). Ci sono due istruzioni POKE con i valori letti dal vettore A. Essi provengono dalla stessa linea della nota di tenore, ma quella di Soprano è nella colonna 1 e quella di Basso nella colonna 3. I valori sono stati scelti in modo che si armonizzino bene con la melodia del Tenore. L'accompagnamento si sente per un intervallo di tempo (linea 70), poi è interrotto per poco (subroutine 1500, dalla linea 190), quindi si ode ancora per un solo periodo (linea 210) e al termine finisce (linea 220).

In seguito cosa accade dipende dal preciso valore di J. Ogni tre note di Tenore si ha una esecuzione per un intervallo doppio di quello normale (fig. 5.5). La linea 230 mostra come sia possibile prevedere quando ciò accade. La nota di Tenore è prolungata quando J è uguale a 3, 6, 9 e 12.

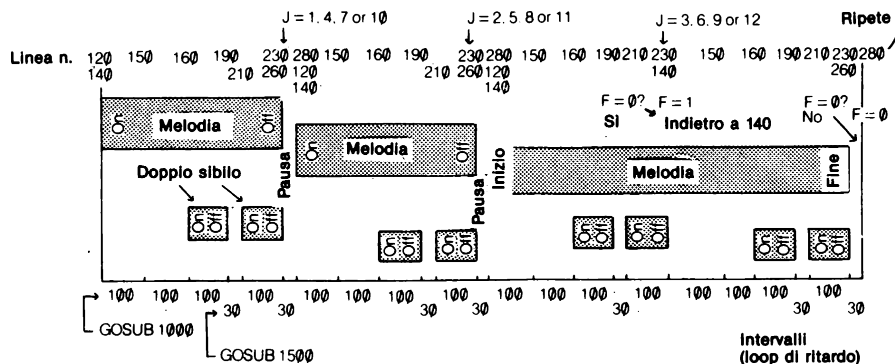


Fig. 5.5 - Carta del suono per il programma della Giostra in funzione del tempo. Il cambiamento della frequenza dipende dalle note del motivo scelto. Il volume è 15, eccetto i periodi di pausa.

Tutti questi valori non danno resto se divisi per tre. Il risultato è un intero, così se dividiamo J per 3 e usiamo la funzione INT sul risultato otteniamo lo stesso numero.

Per esempio,	6/3 = 2	e	INT(6/3) = 2,	
ma	7/3 = 2.3333	e	INT(7/3) = 2.	

Per i numeri divisibili per 3, l'istruzione 230 è vera e il calcolatore ritorna alla linea 140 per ripetere il doppio squillo del Basso e del Soprano. Ma le cose non sono così semplici come potrebbero sembrare. Quando il computer ritorna alla 230 per la seconda volta, come può sapere se deve anda-

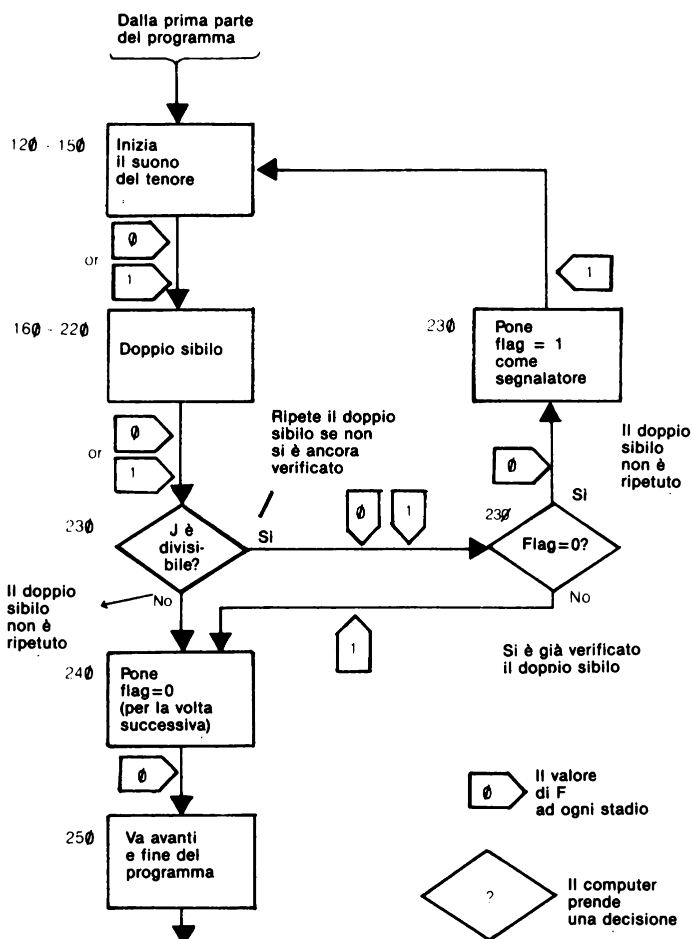


Fig. 5.6 - Come funziona una variabile di flag. I numeri che compaiono sono i numeri delle istruzioni del programma della Giostra.

re ancora indietro alla 140? Il segreto è la variabile F. Le è stato dato questo nome per ricordarci che sta per 'flag' (= bandiera).

Un 'flag' è una variabile posta (generalmente) uguale a 0 o 1 per indicare alla macchina se qualcosa deve o non deve verificarsi. In questo programma F è pari a 1 quando il computer è ritornato indietro alla linea 140 per ripetere il doppio squillo (fig. 5.6). Potete vedere ciò alla linea 230.

Se il calcolatore arriva alla 230 e J non è divisibile per 3, va direttamente alla fine del loop, interrompendo il Tenore e il rumore del motore alla linea 250. Dopo una breve pausa (linea 260) ritorna alla 110 e poi esegue la nota successiva.

Se alla 230, J è divisibile per 3 e flag è zero, si ritorna alla 140 per prolungare la nota e ripetere il doppio squillo. Ma dopo la linea 230 F è posta a 1.

Quando ritorna alla 230, J è ancora divisibile per 3, ma questa volta F = 1. Così l'istruzione alla linea 230 non è completamente vera. Per il computer 'non completamente vera' significa 'falsa' e così continua alla 240. Qui F è azzerata, pronta per essere usata la volta successiva quando J sarà ancora divisibile per 3. Questo è l'impiego tipico di una variabile flag.

Dopo l'esecuzione dell'intero motivo, la sequenza è ripetuta indefinitamente a causa del 'GOTO 110' della linea 280.

Anche a colori!

La Giostra è così carina che è un vero peccato non avere qualche immagine piacevole sullo schermo. Introducete queste nuove linee:

```
55 PRINT"♥":POKE36879,140:PRINT:FORJ=1TO11
56 PRINT"█♥█X";:NEXTJ
57 PRINT:PRINT:PRINTTAB(7)"█C█A█R█O█U█S◣E█L"
135 POKE36879,26+J*16+INT(RND(1)*6)
READY.
```

Questa volta l'effetto è decisamente diverso! La linea 55 cancella tutto ciò che c'è sul video (Vi ricordate come ottenere il 'cuore'? Premendo SHIFT con CLR/HOME). In questo modo lo schermo è color porpora con il bordo arancione. Al termine stampa ripetutamente una configurazione di cuori e di croci. I primi sono rossi e i secondi blu. Per introdurre queste linee, avete bisogno di battere (dopo PRINT):

CTRL assieme a 3, per ottenere la stampa in rosso: appare in reverse la lettera '£'. Questo comando non ha un effetto immediato. I caratteri appaiono ancora blu per il momento, ma diventeranno rossi non appena sarà lanciato il programma.

Poi premete SHIFT con S per ottenere il cuore.

Quindi CTRL con il 7 per la stampa in blu. Ciò produce una freccia a sinistra.

Infine SHIFT con V produce una grossa croce.

Al termine battete ' per chiudere l'istruzione PRINT. Un metodo simile è usato nella linea 56 per far apparire 'Carousel (La Giostra)', dove ogni lettera ha un colore diverso. Prima di scriverle, decidete il colore e premete CTRL con uno dei tasti dei numeri.

```
10 DIM N(12),A(8,3)
20 S1=36876:S2=36875:S3=36874
25 S4=36877:V=36878
30 FOR J = 1 TO 12
40 PRINT"NO.";J;:INPUTN(J)
50 NEXT.J
55 PRINT"▯":POKE 36879,140:PRINT:FOR J=1TO11
56 PRINT"▒▄▀x";:NEXT J
57 PRINT:PRINT:PRINTTAB(7)"▒C█A▒R▒O▒U█S▄E▒L"
60 FOR J = 1 TO 8
70 FOR K = 1 TO 3
80 READ A(J,K)
90 NEXT K
100 NEXT J
110 FOR J = 1 TO 12
120 POKE S2,A(N(J),2):POKES4,253
130 POKE V,15
135 POKE 36879,26+J*16+INT(RND(1)*6)
140 GOSUB 1000
150 GOSUB 1000
160 POKE S1,A(N(J),1:POKE S3,A(N(J),3)
170 GOSUB 1000
180 POKE S1,0:POKE S3,0
190 GOSUB 1500
200 POKE S1,A(N(J),1):POKE S3,A(N(J),3)
210 GOSUB 1000
220 POKE S1,0:POKE S3,0
230 IF J/3=INT(J/3)AND F=0THEN F=1:GOTO140
240 F = 0
250 POKE S1,0:POKE S4,0
260 GOSUB 1500
270 NEXT J
280 GOTO 110
1000 FOR K=1 TO 100:NEXT K:RETURN
1500 FOR K=1 TO 30:NEXT K:RETURN
2000 DATA 230,193,230,232,200,232,214
2005 DATA 206,222,218,209,230
2010 DATA 232,214,227,230,218,232,214
2015 DATA 222,232,218,224,214
```

Listato 5.7 - L'aggiunta del colore nel programma della Giostra.

La linea 135 muta il colore dello schermo e del bordo in modo casuale, a tempo di musica. La strana espressione usata esclude i colori nero e bianco. Non vogliamo nulla di tristemente bianco o nero quando la Giostra inizierà a suonare!

Riassunto

In questo capitolo vi è stato spiegato come:

- usare i generatori del suono del VIC e il controllo del volume
- usare INPUT per introdurre due variabili allo stesso tempo
- memorizzare le informazioni nelle istruzioni DATA e leggerle mediante READ all'interno del programma
- usare il comando RESUME per leggere i dati dall'inizio dell'istruzione DATA
- usare loop FOR... NEXT nidificati
- dimensionare i vettori, introdurvi dati e leggerli
- usare una variabile come flag
- fare in modo che il VIC stampi lettere in colori differenti

Capitolo 6
L'organizzazione della macchina

Interrompiamo per un attimo la descrizione delle musiche e dei gradevoli colori della Giostra per analizzare in che modo esattamente il VIC esegue ciò che gli viene ordinato. Avete già usato alcuni dei comandi descritti in questo capitolo, ma vi sarà spiegato il loro scopo in tutti i dettagli. Vi sono anche alcune nuove parole chiave del BASIC che saranno usate nei successivi capitoli.

Ripetizioni

I calcolatori sono molto abili nelle ripetizioni di azioni. A differenza della maggior parte delle persone, non si annoiano nel dover fare la stessa cosa per diverse volte. Per realizzare ciò battete il programma mostrato nel Listato 6.1.

```
10 FOR J = 1 TO 10000
20 PRINT J;"CHE NOIA!"
30 NEXT J
```

Listato 6.1 - Il programma più noioso che abbiate mai scritto (forse!).

Lanciate il programma. Se vi annoiate prima che il computer finisca, dovete solo premere RUN/STOP. Bene! dopo aver fatto il punto della situazione, andiamo avanti.

Di fatto il calcolatore non esegue esattamente la stessa cosa ogni volta che finisce un loop, perchè il valore di J è stato incrementato. Ogni volta

```
10 PRINT"♥"
20 INPUT"QUALE TABELLINA";N
30 FOR J = 1 TO 10
40 PRINT J;"PER";N;"=";J*N
50 NEXT J
```

Listato 6.2 - Tabelline moltiplicative.

viene stampato un numero diverso, ma anche contare da 1 a 10000 rimane sempre un compito noioso. Il programma del Listato 6.2 usa il comando FOR...NEXT per realizzare qualcosa di più utile.

Quando lanciate il programma, vi viene chiesto 'QUALE TABELLINA?'. Il calcolatore stamperà una tabellina della moltiplicazione, e vuole sapere quale desiderate. Introducete un numero (qualunque, anche negativo) e premete RETURN. Compare il risultato. Alla fine di ogni loop J è incrementata di 1 e viene stampata una linea della tabellina. Ora modificate la linea 30 del programma come segue:

```
30 FOR J = 1 TO 10 STEP 2
```

Il comando STEP incrementa J di 2 ogni volta, invece che di 1. Ora la tabella consiste solo nelle linee 'per 1', 'per 3', 'per 5', 'per 7' e 'per 9'. Dopo di che J arriva a 11 ma poichè è maggiore del massimo valore consentito dal comando 'J = 1 TO 10' il loop non è ripetuto. Come dovrebbe essere la linea 30 per stampare le linee 'per 2', 'per 4', ..., 'per 10'? Introducete quella che pensate sia la linea corretta e verificate se funziona. Ora provate a far stampare al calcololatore le linee 'per 1', 'per 4', 'per 7' e 'per 10'.

Per far stampare le tabelline alla rovescia, battete:

```
30 FOR J = 10 TO 1 STEP —1
```

Cosa accade se togliete 'STEP —1'? Il computer stampa la linea 'per 10'e poi si ferma. Ciò significa che passa attraverso il loop almeno una volta, anche se l'istruzione non ha senso.

Non è necessario che il numero che segue STEP sia un intero. Provate:

```
30 FOR J = 1 TO 2 STEP .1
```

oppure:

```
30 FOR J = —4 TO 4 STEP .5
```

Non c'è alcun limite alle tabelle moltiplicative che possono essere calcolate. Per finire, provate questo:

```
30 FOR J = 1E2 TO 1E3
```

Potete vedere che gli esponenziali sono ammessi nei loop FOR...NEXT. Dimostra anche come gli esseri umani si annoino più velocemente delle

macchine. Ancora una volta usate il tasto RUN/STOP vista la inutilità di questo programma.

Come accennato nel capitolo cinque, i loop FOR...NEXT possono essere nidificati uno nell'altro. Dovete usare come contatori di ogni loop variabili diverse, come nel programma del Listato 6.3, dove i contatori sono J nel loop più esterno e K in quello interno.

```
10 PRINT"J"
20 INPUT"QUALE TABELLINA?";N,M
30 FOR J = 1 TO N
40 FOR K = 1 TO M  )               )
50 PRINT J*K;      } loop interno  ( loop esterno
60 NEXT K          )               (
70 PRINT""                         )
80 NEXT J
```

Listato 6.3 - Altre tabelle moltiplicative.

Questo programma stampa più tabelle nello stesso tempo, come potete vedere quando lo lanciate. Ora i numeri che devono essere introdotti sono due. Sarebbe meglio che K non fosse maggiore di 6, in caso contrario ogni tabella occupa più di una linea e il risultato sullo schermo apparirebbe confuso.

Per ogni valore di J, il loop interno stampa un gruppo di valori di J moltiplicato per K, 'la tabella dei multipli di J', da 1 a K. L'istruzione PRINT della linea 50 è seguita da un punto e virgola. Ciò fa in modo che il computer stampi le risposte sulla stessa linea dello schermo. Quando ha finito una tabella, termina il loop interno e passa alla linea 70. Qui stampa una stringa nulla (cioè niente). Notate che questa istruzione non finisce con un punto e virgola. La sua funzione è solamente quella di far passare il computer alla riga successiva dello schermo.

È possibile nidificare più di due loop uno nell'altro. Per esempio, potreste avere il programma mostrato nel Listato 6.4.

```
10 FOR J = 1 TO 10
20 FOR K = 1 TO 100
30 FOR L = 1 TO 1000
40 FOR M = 1 TO 10000
50 PRINT"ANCORA PIU' NOIOSO!"
60 NEXT M
70 NEXT L
80 NEXT K
90 NEXT J
```

Listato 6.4 - Loop nidificati in un programma ancora più noioso di quello del Listato 6.1.

Questi loop sono nidificati, essendo ognuno completamente inserito nel successivo. Partiamo con J, K, L e M, e terminiamo nell'ordine opposto: M, L, K, J. Dopo aver stampato il messaggio diecimila milioni di volte,

il calcolatore si prende un meritato riposo.

Come abbiamo visto, i valori che compaiono in un loop FOR...NEXT possono essere costanti (FOR J = 1 TO 10), variabili (FOR J = N TO M) od espressioni (FOR J = 2*M/3 TO 4*(N+1)+23). Ciò che è importante osservare è che nessuno di questi valori può essere cambiato all'interno dei loop. Per esempio, se N o M sono ricalcolati all'interno del loop, il programma può fallire completamente. Se volete cambiarne i valori mentre il loop è in esecuzione, usate il metodo mostrato nel Listato 6.5.

```
10 PRINT"♥"
20 INPUT"IL MASSIMO";M
30 N = 0
40 J = 1
50 N = N + J
60 PRINTN
70 IF N < M THEN J = J + 1: GOTO 50
80 END
```

Listato 6.5 - Costruzione di un loop in cui il massimo valore del contatore non è definito a priori (l'equivalente di REPEAT ... UNTIL).

Il diagramma di flusso (fig. 6.1) spiega come lavora il programma. Esso stampa una serie di numeri: 1, 3, 6, 10, 15, 21, ecc. La differenza tra un numero e il successivo aumenta sempre di uno:

I numeri:	1		3		6		10		15		21
Le differenze:		2		3		4		5		6	

All'inizio del programma vi viene chiesto di fissare un valore massimo. J è il contatore del loop. Non è possibile prevedere in anticipo quante volte sarà eseguito il loop. Ciò significa che non è possibile usare un FOR...NEXT come al solito. Infatti il programma calcola e stampa ogni numero e quindi verifica se ha terminato. In caso contrario, torna indietro e stampa il numero successivo. Non abbiamo usato un comando speciale come FOR...NEXT, ma abbiamo costruito il loop mediante altri comandi.

Nella linea 70 di questo programma appare un nuovo simbolo. 'IF N < M' significa 'Se N è minore di M'. In altre parole il loop si ripete fino a che i numeri stampati sono più piccoli del massimo che avete fissato. Quando N è uguale a M o diviene maggiore, questo valore finale di N viene visualizzato ma il loop non è ripetuto e il programma finisce.

Il tempo

Molto spesso si usa un ciclo FOR...NEXT per realizzare un ritardo nel-

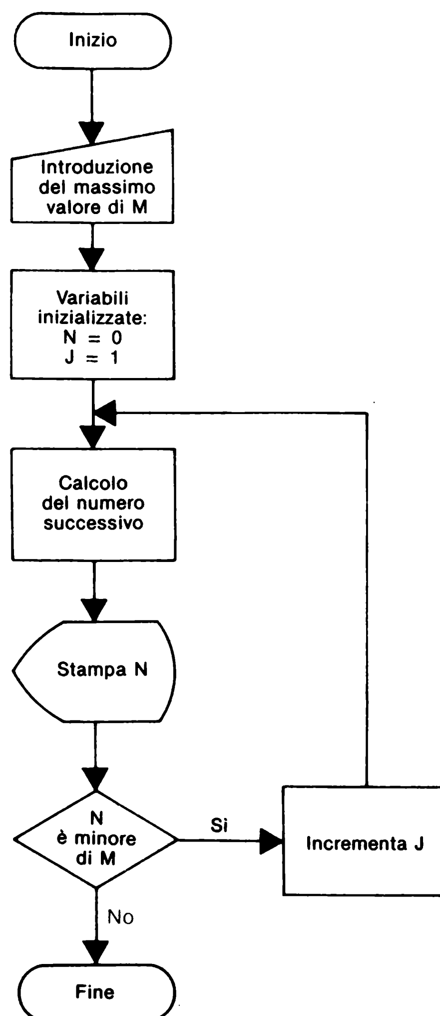


Fig. 6.1 - Schema di flusso per il programma del Listato 6.5.

l'esecuzione del programma. Abbiamo mostrato alcuni esempi in precedenza in cui il computer passava attraverso un loop del tipo 'FOR K = 1 TO 3000: NEXT K' in modo da impedire alla macchina di passare immediatamente al comando successivo. Questo è un mezzo molto utile per ottenere dei ritardi di diverse ampiezze. Per contare fino a 1000 il calcolatore impiega circa 1 secondo, perciò un loop di '1 TO 3000' fornisce un ritardo di 3 secondi. Il programma del Raid Aereo nel Listato 5.4 è un esempio di come si possa realizzare un ritardo di durata casuale.

Se invece si vuole che quest'ultima sia di lunghezza prefissata, si puo

usare la variabile TI. In un programma di un gioco, per esempio, può essere necessario riservare a ciascun giocatore uno spazio di tempo fisso per ogni mossa. Il programma mostrato nel Listato 6.6 concede a ciascun giocatore due minuti, quindi fa comparire il messaggio 'IL TEMPO È SCADUTO':

```
10 S = 120
20 TI$ ="000000"
30 IF TI < S * 60 THEN GOTO 30
40 PRINT"IL TEMPO E' SCADUTO"
```

Listato 6.6 - Un programma per il tempo scaduto

La variabile S è la lunghezza di tempo richiesta, espressa in secondi. Alla linea 20 l'orologio del VIC è inizializzato a zero. Porre TI$ uguale a '000000' significa azzerare anche TI. Quest'ultima è incrementata 60 volte al secondo. Il computer ripete la linea 30 fino a che TI non è uguale o maggiore di 2700, quindi stampa il messaggio. Un programma di questo tipo è meglio utilizzato come subroutine. Per fare in modo che il programma realizzi un ritardo di tempo di qualsiasi durata, cambiate la linea 10 in:

10 INPUT 'QUANTI SECONDI';S

Ora non rimarrete mai senza un temporizzatore da cucina ad alta precisione!

Come si prendono le decisioni

Quando il computer arriva a una istruzione del tipo:

IF (condizione) THEN (azione)

deve decidere se la condizione è vera o falsa. La condizione può essere una semplice ugualianza, ad esempio:

60 IF X = 10 THEN GOTO 120

Se la condizione (X = 10) è vera, cioè se X è realmente pari a 10, la macchina esegue l'azione specificata. In questo caso salterebbe dalla linea 60 alla 120, continuando il programma da quel punto. Se invece la condizione è falsa (cioè X ha un valore diverso da 10) l'azione non è eseguita. Il calcolatore non fa nulla e passa alla istruzione successiva alla 60.

Il segno di uguale nella condizione esaminata è uno dei diversi operatori relazionali. Esso specifica una relazione fra due quantità, che in questo caso sono il valore di X e 10. Ecco altri due esempi di operatori relazionali:

< che significa 'minore di'
> che significa 'maggiore di'

Potete combinare assieme due operatori relazionali:

X <= 10 significa 'X è minore o uguale a 10'
X >= 10 significa 'X è maggiore o uguale a 10'
X < > 10 significa 'X è minore o maggiore di 10'

L'ultimo caso equivale a dire 'X è diverso da 10'. C'è un programma nel Listato 6.7 che mostra come funziona tutto ciò.

```
10 PRINT"♥"
20 INPUT"QUALE E' LA TUA ETA'";A
30 PRINT
40 IF A<0 THEN PRINT"NON ESSERE SCIOCCO!"
50 IF A<=5 THENPRINT"SEI GIOVANE PER LEGGERE"
60 IF A=5 THENPRINT"HAI INIZIATO LA SCUOLA?"
70 IFA>5ANDA<17THENPRINT"TI DIVERTI A SCUOLA?"
80 IF A<>21 THENPRINT"VORRESTI AVERE 21 ANNI?"
90 IF A>12ANDA<20 THENPRINT"CIAO,TEENAGER!"
100 IFA=>20THENPRINT"COSA TI PIACEREBBE";A;"?"
110 IF A=>80 THENPRINT"CONGRATULAZIONI!"
120 IF A > 100 THENPRINT"???"
130 END
```

Listato 6.7 - Quiz sull'età, un programma che mostra come usare gli operatori relazionali.

Il suo nome è 'quiz sull'età'. Introducete la vostra età e il computer vi fornirà una risposta appropriata. Per esempio, se la vostra età è 17 anni, le condizioni delle linee 40, 50, 60 e 70 sono false. Il computer passa oltre a queste istruzioni senza fare nulla. Quando arriva alla linea 80 trova che la condizione è vera - la vostra età non è 21 anni. Il calcolatore passa oltre alla 80 e stampa 'NON VORRESTE AVERE 21 ANNI?'.

Anche la condizione nella linea 90 è verificata, perchè la vostra età è maggiore di 12 e minore di 20. Così compare il messaggio 'CIAO, TEENAGER!'. Da questo punto in poi il calcolatore non trova più alcuna condizione vera, così non stampa più nulla.

I commenti forniti dal programma sono del tutto generali, ma potreste

divertirvi rendendoli più adatti ai vostri familiari.

Le condizioni nelle linee 70 e 90 includono la parola chiave di BASIC AND. Essa agisce esattamente come in una normale frase italiana. La condizione è vera solo se entrambe le sue due parti lo sono. Se avete 17 anni, la vostra età è maggiore di 5, ma non è minore di 17. Nella linea 70 solo una delle due condizioni è vera, perciò l'intera istruzione è falsa. D'altra parte, nella linea 90 la vostra età è maggiore di 12 ed è minore di 20, pertanto l'istruzione nella sua totalità è vera e perciò appare il messaggio di saluto al teenager.

Anche se AND appare come una parola ordinaria, essa è molto più importante. È uno dei tre operatori relazionali che la macchina è in grado di capire. Gli altri due sono OR e NOT. OR unisce due condizioni, ma è necessario che solo una di esse sia vera. Potreste aggiungere questa linea:

```
95 IF A <13 OR A> 19 THEN PRINT 'NON SEI UN TEENAGER'
```

Se avete 11 anni, la prima condizione è vera, la seconda è falsa. Tuttavia OR richiede che una sola delle due sia vera, così la condizione è verificata e il messaggio viene stampato. Analogamente accade se avete 35 anni, perchè la prima condizione è falsa mentre la seconda è vera. Tuttavia se siete un teenager entrambe le condizioni sono false, come l'intera istruzione e il commento non appare.

L'operatore NOT rende falsa una condizione che altrimenti (senza NOT) sarebbe vera. Per esempio, potete modificare la linea 40:

```
40 IF NOT A = > 0 THEN PRINT 'NON ESSERE SCIOCCO!'
```

Questa ha lo stesso effetto della linea 40 preesistente, quindi non c'è nulla di interessante in questa variazione. Tuttavia vi sono esempi in cui l'uso di NOT rende più semplice capire la logica dell'espressione (probabilmente la linea vi sembrerà più sensata con questa forma).

Potete combinare assieme più AND, OR o NOT includendo in un'istruzione diverse condizioni. Il programma 'Animali' del Listato 6.8 vi mostra come fare.

L'idea è semplice. Pensate un animale e rispondete alle domande. Esse sono solamente tre (vedere le istruzioni 30 e 50). Il computer cerca di indovinare quale animale avete pensato. Naturalmente se volete avere risposte attendibili dovete scegliere un animale che il computer conosce (vedere le linee dalla 70 alla 120).

Dovete dire al calcolatore il numero di zampe (L) di ali (W) e se l'animale può volare. Se può volare F = 1, in caso contrario F = 0. Dalla istruzione 70 alla 120 la macchina esegue operazioni logiche su L, W e

```
10 PRINT"♥"
20 PRINT"CHE COS'E'?":PRINT
30 INPUT"QUANTE ZAMPE";L
40 INPUT"QUANTE ALI";W
50 INPUT"PUO' VOLARE (S/N)";F$
60 IF F$="S" THEN F = 1
70 IF L=2 AND W=0 AND F=0 THEN A$="ASINO"
80 IF L=2 AND W=2 AND F=0 THEN A$="KIWI"
90 IF L=4 AND W=0 AND F=0 THEN A$="MUCCA"
95 IF L>20 AND W=0 AND F=0 THENA$="MILLEPIEDI"
99 IF(L=4 OR L=2)AND W=2 AND F=1 THENA$="GUFO"
120 IFL=0ANDW=0ANDF=0 THEN A$="FALENA"
125 IFL>5ANDL<20ANDW=0ANDF=0 THEN A$="FALENA"
126 IFL=6ANDW=4ANDF=1 THEN A$="FALENA"
130 PRINT"♥"
140 IF A$="" THEN GOTO 170
150 PRINT"POTREBBE ESSERE";A$;"."
160 END
170 PRINT"NON CONOSCO QUESTO ANIMALE."
180 END
```

Listato 6.8 - Il programma degli Animali mostra come usare gli operatori logici.

F per scoprire il nome dell'animale. La linea 70 afferma che se ha quattro zampe, non ha ali e non può volare si tratta di un asino. Qui abbiamo usato tre AND, quindi devono essere vere tutte e tre le condizioni affinchè il computer stampi 'ASINO'. Una simile logica è applicata nelle linee 80 e 90. Le istruzioni dalla 70 alla 90 potrebbero essere semplificate eliminando 'AND F = 0' da ognuna di esse, ma ciò creerebbe maggiori difficoltà se si decidesse di aggiungere un altro animale nel programma, per esempio un passero.

La linea 100 permette all'utente di introdurre un numero qualsiasi più grande di 20.

Qualcuno potrebbe dire che un pipistrello ha 2 zampe oppure 4, dipende da ciò che si intende per 'zampa'. L'istruzione 110 vi permette di inserire sia 4 che 2 come numero di zampe. Osservate questa linea. Potreste pensare che sarebbe più corretto scrivere un'istruzione come questa:

110 IF L = 4 OR L = 2 AND W =2 AND F = 1 ...

Ciò non dà lo stesso risultato perchè il computer esegue prima tutti gli AND, poi gli OR. In altre parole esiste una regola di priorità, proprio come per gli operatori aritmetici (vedere capitolo quattro). Senza le parentesi l'istruzione esatta sarebbe:

L = 4 (non importa quale valore abbia W o F)
OR L = 2 AND W = 2 AND F = 1

In questo modo qualsiasi animale a quattro zampe (per es. una mucca) avrebbe come risposta un 'PIPISTRELLO'. La regola di priorità fa in modo che il computer esegua prima le espressioni tra parentesi. Le parentesi nella linea 110 costringono il calcolatore ad operare prima sull'OR. Se L = 4 OR L = 2, l'espressione tra parentesi, è vera (i pipistrelli hanno 2 o 4 zampe). Perciò se è vera E W = 2 E F = 1, l'animale in questione è un pipistrello.

La linea 120 racconta la storia della complessa vita di una falena. Da adulta ha 6 zampe, 4 ali e può volare, mentre quando è ancora un bruco ha 6 vere zampe ed altre appendici simili ad esse, non può volare ed è senza ali. Quando diventa una crisalide non ha zampe, ali e non può volare. Il computer deve analizzare tutte e tre le possibili combinazioni di caratteristiche:

senza zampe, senza ali, e non può volare (la crisalide)
OR tra 6 e 19 'zampe' (dipende da ciò che intendete per 'zampa'), senza ali e non può volare (il bruco)
OR 6 zampe, 4 ali e può volare (falena adulta)

Se si verifica una di queste tre combinazioni, l'animale è una falena. I vari gruppi di caratteristiche sono legati assieme da AND per verificare se i dettagli introdotti si adattano a uno degli stadi della vita di una falena. Notate che non c'è alcun bisogno di parentesi nella linea 120 perchè AND ha la priorità su OR.

Il calcolatore può organizzare la sua logica in un altro modo che troverete più semplice da usare nella programmazione. Il programma del Listato 6.8 chiedeva 'Quante zampe?', e 'Quante ali?' e poi lavorava su numeri che potevano avere valori diversi. Potremmo semplificare la logica se rivolgessimo domande che prevedono solo due possibili risposte - 'Si' o 'No' oppure 'Vero' o 'Falso'. Le variabili corrispondenti assumono il valore 0 se la risposta è 'No' (o 'Falso') e —1 se la risposta è 'Si' (o 'Vero'). Il computer comprende questo metodo, pertanto se per esempio poniamo A = —1 e B = —1 alla linea:

```
10 IF A AND B THEN PRINT 'VERO'
```

otteniamo 'VERO'. La condizione è vera se entrambe A e B lo sono (entrambe sono '—1'). Invece se A e B hanno un'altra coppia di valori (0,0 o 0,—1 o —1,0) A AND B risulta falso e il messaggio non viene stampato. Il Listato 6.9 è simile al programma che indovina l'animale, l''Animalogic', e mostra come si può usare questa logica. Certamente rende più semplice la comprensione della logica del programma.

```
10 PRINT"♥"
20 PRINT"CHE COS'E'?"
30 INPUT"HA ALI";W$
40 IF W$="S" THEN W=-1
50 INPUT"PUO' VOLARE";F$
60 IF F$="S" THEN F=-1
70 INPUT"HA UNA PELLICCIA";C$
80 IF C$="S" THEN C=-1
90 IF W AND F AND C THEN A$="PIPISTRELLO"
100 IF NOT W AND F AND NOT C THENA$="RAGNO"
110 IF W AND NOT F AND C THENA$="OSTRICA"
120 IFNOT W AND NOT F ANDNOT C THENA$="FALENA"
125 IF W AND F AND NOT C THENA$="FALENA"
130 PRINT"♥"
140 IFA$="" THEN GOTO 170
150 PRINT"POTREBBE ESSERE";A$
160 END
170 PRINT"NON CONOSCO QUESTO ANIMALE."
180 END
```

Listato 6.9 - Il programma Animalogic è un altro esempio di programmazione logica.

In questo programma tutte le domande richiedono come risposta Si/No, pertanto battete 'S' o 'N' per ognuna di esse. Alla linea 90 l'istruzione è vera se lo sono tutte e tre W, F e C, cioè se hanno tutte valore '—1'. La linea 100 esclude tutti coloro che vogliono prendersi gioco del calcolatore. Un ragno non ha ali, ma può volare per miglia, attaccato al filo sottile della ragnatela! Esso non è ricoperto da una pelliccia, quindi C è falso, oppure detto in un altro modo, NOT C è vero. La linea 110 è un animale con ali (W è vera), che non può volare (NOT F è vera), e non ha una pelliccia (NOT C è vera). Infine alla 120 abbiamo la logica più complicata della falena. All'inizio consideriamo il caso del bruco e della crisalide assieme (niente ali, niente pelliccia, non può volare). Poi dopo l'OR, il caso adulto (con ali, senza pelliccia, può volare). Il computer elabora prima gli AND, che stanno a destra e a sinistra dell'OR. Infine se gli OR di due di queste condizioni AND si adattano alle caratteristiche della falena, la macchina pone A$ uguale a 'FALENA'.

L'impiego dei numeri nelle decisioni

La parola BASIC 'ON' ci consente di avere una specie di 'segnaposto' riferito a diverse parti del programma. La sintassi del comando è:

> ON N GOTO ...
> o ON N GOSUB..

Invece di N potete porre qualsiasi altra variabile numerica o intera o un'e-

```
10 PRINT"⌐"
20 INPUT"IL VOSTRO TURNO";X
30 N = INT (RND(1)*6) +1:PRINT
40 ON N GOTO 100,200,300,400,500,600
100 PRINT"PERDETE 1"
110 GOSUB 1000
120 GOTO 10
200 PRINT"▒VINCETE 1"
210 GOSUB 1000
220 PRINT"▀":GOTO 10
300 PRINT"■PERDETE 3 - CHE SFORTUNA"
310 GOSUB 1000
320 PRINT"▀":GOTO 10
400 PRINT"▒VINCETE 3 - HURRA'!"
410 POKE 36879,90
420 GOSUB 1000
430 PRINT"⌐:GOTO 10
500 PRINT"■PAGATE LA POSTA 2"
510 POKE 36879,24
520 GOSUB 1000
530 PRINT"⌐":GOTO 10
600 PRINT"▒VINCETE ▌LA ▒POSTA"
610 FOR K = 0 TO 6
620 POKE 36879,153 + K
630 GOSUB 1000
640 NEXT K
650 GOTO 10
1000 FOR J = 1 TO 2000:NEXT J
1010 RETURN
```

Listato 6.10 - Put and Take, un gioco d'azzardo basato sulla generazione di numeri casuali.

spressione aritmetica. Il programma nel Listato 6.10 mostra come usare l'istruzione GOTO.

Un aspetto interessante di questo programma è l'impiego della funzione RND per produrre un numero casuale tra 1 e 6, come quando lanciate un dado. Quando siete pronti, premete RETURN. Il valore X è una variabile muta che non viene usata nel programma. Il numero casuale è N e, una volta giunti alla linea 40, il computer salta a una delle linee elencate mediante l'istruzione GOTO. Per esempio, se N vale 3, salta alla terza linea indicata, cioè 300. In questa parte del programma vi viene detto cosa fare se avete ottenuto 3 ai dadi. Notate che in questo programma i numeri delle linee sono centinaia pari, e ciò serve per renderlo più semplice e leggibile. Tuttavia in un altro programma potete avere i numeri di linea in un ordine qualsiasi e non necessariamente centinaia pari:

```
100 ON Z GOTO 123, 456, 321, 6, 777
```

In questo esempio quando Z vale 3 il computer salta alla linea 321.

Questo programma è semplice ma realizza un gioco d'azzardo per due giocatori. Potete facilmente modificarlo per ottenere un gioco a voi più familiare. Vi sono diversi numeri di linea da utilizzare nelle varie parti del programma, quindi si può migliorare l'aspetto di ciò che appare sullo

schermo. Potete anche aggiungere alcuni effetti sonori. Il programma FXG nel Listato 9.1 del capitolo nove vi aiuterà in questo intento.

Riassunto

In questo capitolo vi è stato spiegato come:

— programmare con loop FOR...TO...STEP...NEXT
— nidificare i loop in modo corretto
— fare ripetere il programma fino a che una condizione non è verificata
— usare ON...GOTO e ON...GOSUB
— usare la variabile TI
— usare gli operatori relazionali, = < e >
— scrivere programmi con diramazioni, usando IF (condizione) THEN (azione)
— usare gli operatori logici, AND, OR e NOT, per collegare sia due o più istruzioni condizionali, sia due o più variabili che hanno un valore logico.

Avete anche imparato che:

— una variabile con valore '0' ha un significato logico di 'Falso'
— una variabile con valore '—1' ha un significato logico di 'Vero'
— la regola di priorità tra operatori logici è:
  Per primo, le espressioni tra parentesi
  Poi, AND
  Quindi, OR
  Infine, NOT.

Capitolo 7
La grafica

Esistono tre modi di programmazione del VIC per ottenere delle immagini colorate sullo schermo:

(1) usare l'istruzione PRINT
(2) usare l'istruzione POKE nella RAM del video
(3) usare la funzione CHR$

Abbiamo già impiegato i primi due metodi, mentre il terzo è nuovo. È quello generalmente meno usato, forse perchè gli altri due sono più convenienti, tuttavia è utile conoscere anche questa terza funzione.

Tutte le lettere, i numeri, i simboli e gli altri caratteri che il computer può usare hanno un numero di codice che li identifica. Questo è detto codice ASCII, dalle iniziali di American Standard Code For Information Interchange. Originariamente questi codici venivano usati per la trasmissione di dati tra telescriventi. La funzione CHR$ usa il codice ASCII nel modo seguente:

```
10 PRINT CHR$(65)
```

Quando lanciate questo breve programma, il calcolatore stampa semplicemente 'A' perchè, come mostra la tabella 10 (Appendice A), il codice ASCII di 'A' è 65. Sembra un modo piuttosto elaborato per stampare 'A', dal momento che potreste semplicemente scrivere:

```
10 PRINT 'A'
```

Tuttavia questo metodo presenta dei vantaggi. Può accadere che dobbiate stampare un codice dei caratteri, ma che la scelta sia fatta durante l'esecuzione del programma. Allora potreste avere una linea del tipo:

```
250 PRINT CHR$(N + 6)
```

Il carattere visualizzato dipende dal valore di N.

Il codice ASCII fu realizzato parecchi anni fa, e potete rendervene conto considerando i codici di controllo. Un esempio è CHR$(13), che corrisponde a un 'carriage return' (= ritorno del carrello). Ottenete lo stesso risultato premendo il tasto RETURN. L'uso dei codici di controllo non fa apparire nulla sullo schermo, ma interessa direttamente le operazioni della macchina (telescrivente o computer). Eccetto il carattere CHR$(13), gli altri codici di controllo usati dal VIC non fanno parte dei codici standard ASCII. Essi controllano il cursore, il colore dello schermo, e altre svariate funzioni attraverso la tastiera. Mediante CHR$ potete stampare anche caratteri grafici, tuttavia poichè questi sono propri solamente dei computer Commodore, i loro codici non fanno parte dello standard ASCII.

Come usare l'istruzione POKE

Questo metodo è stato oggetto del capitolo tre, dove abbiamo mostrato parecchi esempi. I numeri che devono essere memorizzati nella RAM del codice dei caratteri mediante POKE sono esattamente i codici ASCII, se state usando i numeri, i simboli e le lettere maiuscole del modo di testo. Le lettere minuscole non hanno un codice ASCII, mentre i caratteri grafici sono memorizzati in un modo complicato. Quando usate l'istruzione POKE è più semplice fare riferimento alla Tabella 6 dell'Appendice A, piuttosto che utilizzare il codice ASCII.

Come usare l'istruzione PRINT

Abbiamo trovato abbastanza esempi dell'impiego di PRINT (linea 55 del programma della Giostra, Listato 5.7). Ora analizzeremo questo metodo più dettagliatamente, perchè è uno dei modi migliori per creare delle figure sullo schermo. Per fare questo consideriamo il programma del Listato 7.1, chiamato Locomotiva.

Esso inizia come al solito cancellando lo schermo e ponendo il cursore

```
10 PRINT"■":POKE 36879,156
20 N = 7
30 PRINT TAB(N)"    ■▒"
40 PRINT TAB(N)"     ▒"
50 PRINT TAB(N)"■■ ■   ▒■ "
60 PRINT TAB(N)" ■■      "
70 PRINT TAB(N)"■       ◥"
80 PRINT TAB(N)"■■ ◠ ◠ "
90 PRINT TAB(N)" ◡ ◡ "
```

Listato 7.1 - La locomotiva.

in alto a sinistra. Ricordate che il 'cuore' stampato nella linea 10 non appare sul video perchè è stato posto nell'istruzione PRINT premendo SHIFT e CLR/HOME. Esso appare nel listato come un cuore, ma non ha un effetto immediato quando lo introducete, o quando visualizzate il programma. Invece quando lanciate il programma è interpretato come SHIFT e CLR/HOME. In altre parole è un carattere di controllo.

Anche la linea 10 cambia il colore dello schermo in arancione con il bordo porpora, in modo da dare uno sfondo adatto alla figura. Quest'ultima è visualizzata nelle linee comprese tra 30 e 90 come un gruppo di sette stringhe di caratteri. Per fare in modo che appaia nel centro dello schermo e non a sinistra, si fa seguire al comando PRINT la parola TAB. Essa dice al computer di iniziare a stampare le stringhe dalla colonna numero N, il cui valore è stato posto a 7 alla linea 20. In questo modo la stampa inizia dalla colonna 7 invece che dalla colonna 0.

Le stringhe contengono sia caratteri che devono essere stampati sullo schermo, sia caratteri di controllo (che non appaiono sul video). Per esempio, la linea 30 inizia con tre spazi, seguiti da un carattere di controllo CTRL e '1'. Esso cambia il colore delle lettere visualizzate in nero. Appare immediatamente il carattere sfumato (la 'bandierina' del Commodore con '+'), che rappresenta il fumo che esce dal camino. La stringa successiva è simile alla precedente ma il fumo compare in una colonna più a destra.

La linea 50 è più complicata. La figura 7.1 mostra come è costituita.

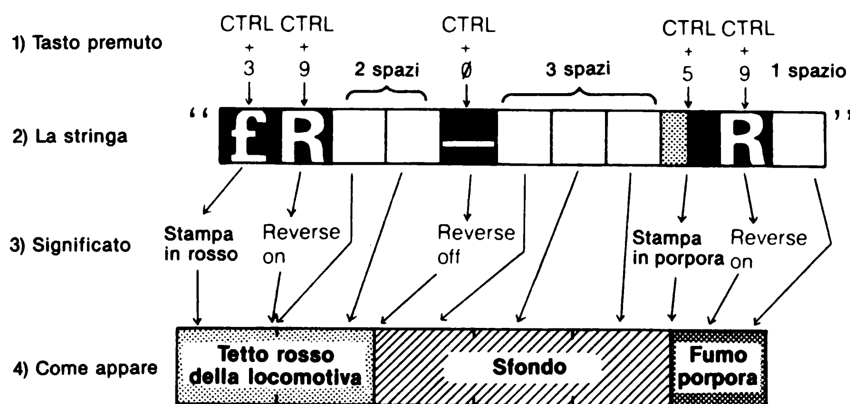


Fig. 7.1 - Realizzazione di figure mediante l'uso dell'istruzione PRINT seguita da stringhe (linea 50 del programma della Locomotiva).

La '£' in reverse significa 'stampa in rosso', mentre la R significa 'stampa in reverse' (si ha lo stesso effetto premendo CTRL e 9). Quindi seguono due spazi vuoti, ma poichè si è in reverse, vengono stampati due quadra-

tini pieni rossi che sono il tetto della locomotiva. Vi sono poi alcuni spazi che hanno lo stesso colore dello schermo. Un carattere di controllo serve per interrompere la stampa in reverse ed è seguito da tre spazi. Notate come abbiamo usato la stampa in 'reverse' per fare apparire quadratini colorati, mentre il comando opposto visualizza i caratteri nel colore dello schermo. Dopo di questo, il colore cambia in rosso porpora. Azioniamo ancora il 'reverse' e stampiamo tre spazi vuoti. In questo modo appare un rettangolo color porpora, per il fumo. Quando lanciate il programma potrete rendervi conto degli effetti degli altri caratteri di controllo delle linee rimanenti.

Un aspetto interessante è che il comando per cambiare colore rimane in azione fino a che il colore non viene nuovamente alterato. Per esempio, nella linea 60 i caratteri sono stampati in rosso, ma non è necessario ripetere il comando nella linea successiva. Al contrario il comando di 'stampa in reverse' è automaticamente disattivato alla fine di ogni istruzione PRINT. È necessario ripristinarlo ogni volta che lo si vuole usare. Nella linea 70 la stringa inizia con 'stampa in reverse', anche se lo stesso comando era presente nella linea precedente. L'unico caso in cui si deve disattivare la stampa in 'reverse' è all'interno di una stringa, come nella linea 50, dopo aver disegnato il tetto della locomotiva.

Cosa ne dite di farla muovere? È un tentativo un po' difficile, ma illustra alcuni punti interessanti. Cambiate la linea 20 del programma come segue:

```
20 FOR N = 0 TO 14
```

Ciò riguarda il valore di N nel comando di TAB, in modo che le stringhe vengono stampate sempre più a destra ogni volta.

Battete queste linee:

```
100 FOR J = 1 TO 150: NEXT J
110 IF N < 14 THEN PRINT '[♥]'
120 NEXT N
```

Lanciate il programma e la locomotiva si muove sullo schermo, fermandosi quando arriva all'estrema destra. Il programma stampa la figura 15 volte in rapida successione, muovendosi verso destra, una colonna alla volta. La linea 100 crea un ritardo che vi consente di visualizzare l'immagine in ogni posizione. La 110 cancella lo schermo e lo prepara per la stampa successiva. L'ultima volta, quando N è uguale a 14, il loop non è eseguito e sul video rimane la figura della locomotiva.

Questo esempio illustra le principali caratteristiche dell'animazione, cioè

la visualizzazione di figure in rapida successione. Le immagini in realtà sono ferme, ma poichè appaiono immediatamente una vicina all'altra, è come se gli oggetti si muovessero. A volte sembra che l'intera figura sia in movimento. È possibile far muovere solo una parte della figura, come sarà spiegato tramite alcuni esempi nel capitolo undici. Nel frattempo, perchè non aggiungete al programma della 'Locomotiva' anche gli effetti sonori?

La memorizzazione dei caratteri

Fino ad ora abbiamo usato caratteri di ogni tipo senza curarci di come il VIC possa riconoscerli e fare apparire le giuste immagini sullo schermo televisivo. Analizziamo questo argomento più in dettaglio. Tra i vari chip, ne esiste uno speciale chiamato generatore di caratteri. Possiamo pensarlo come un gruppo di scatole, ognuna delle quali contiene una parte delle istruzioni che servono per formare un carattere. Alcune di queste devono essere inviate allo schermo televisivo, per ogni carattere visualizzato. Non siamo interessati a capire esattamente come le istruzioni vengono inviate all'apparecchio televisivo, ma vale la pena di analizzare come sono create le istruzioni, in modo che potete disegnare tutti i caratteri che volete. Per esempio, potete convertire alcuni tasti per ottenere una 'alfa' greca o qualsiasi altra lettera di un alfabeto straniero, o un simbolo matematico o scientifico, o anche un'astronave aliena.

Prima di tutto, dove sono memorizzati i caratteri? Il miglior modo per scoprirlo è osservare alcune istruzioni memorizzate nel generatore di caratteri. Avete imparato come introdurre dei dati in una cella di memoria usando l'istruzione POKE. Se invece si vuole accedere al contenuto di una cella di memoria, dovete usare il comando PEEK. Per capire come funziona, battete:

```
POKE 5000,234
```

In questo modo avete memorizzato '234' all'indirizzo 5000, che fa parte della RAM. Ora scrivete:

```
X = PEEK(5000) : PRINT X
```

Notate le parentesi usate dopo la parola PEEK. POKE non impiega le parentesi come PEEK. Sul video appare il risultato della istruzione PEEK: il numero '234' (vedere la fig. 7.2). Ponete un altro numero nello stesso indirizzo, poi usate ancora PEEK per verificare che il numero memorizzato è quello che avete introdotto. Un modo più veloce per fare ciò è:

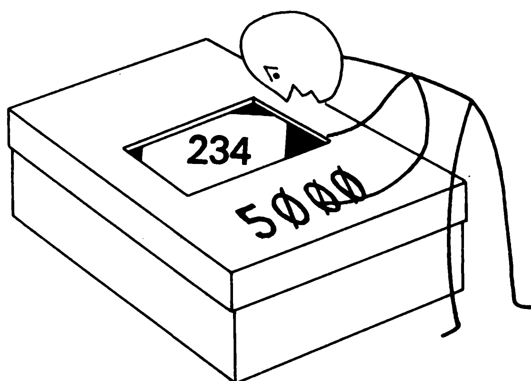


Fig. 7.2 - L'istruzione PEEK legge il valore memorizzato in un indirizzo di una RAM o di una ROM. Confrontate questa figura con la fig. 3.1.

PRINT PEEK(5000)

Ora usiamo l'istruzione PEEK per un indirizzo del generatore di caratteri. Ci sono 4096 indirizzi, che vanno da 32768 a 36863. È un numero molto alto perchè occorrono otto celle di memoria per memorizzare i dati di un singolo carattere. Per sapere qual è il contenuto del primo indirizzo, battete:

PRINT PEEK(32768)

Il risultato è '28'. Se tentate di introdurre in 32768 un qualsiasi altro valo-

Indirizzo	Valore memorizzato (in decimale)	Le potenze di due: 128	64	32	16	8	4	2	1	Le somme	
32768	28	Ø	Ø	Ø	1	1	1	Ø	Ø	16 + 8 + 4	= 28
32769	34	Ø	Ø	1	Ø	Ø	Ø	1	Ø	32 + 2	= 34
3277	74	Ø	1	Ø	Ø	1	Ø	1	Ø	64 + 8 + 2	= 74
32771	86	Ø	1	Ø	1	Ø	1	1	Ø	64 + 16 + 4 + 2	= 86
32772	76	Ø	1	Ø	Ø	1	1	Ø	Ø	64 + 8 + 4	= 76
32773	32	Ø	Ø	1	Ø	Ø	Ø	Ø	Ø	32	= 32
32774	3Ø	Ø	Ø	Ø	1	1	1	1	Ø	16 + 8 + 4 + 2	= 3Ø
32775	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	= Ø
		Valori memorizzati (in binario)									

Fig. 7.3 - Otto numeri binari di otto cifre ottenuti dagli indirizzi successivi del generatore di caratteri.

re, non otterrete alcun risultato, perchè il generatore di caratteri è una specie di ROM. Il numero ivi memorizzato rimane sempre '28'. I primi otto indirizzi del generatore contengono i dati relativi al primo carattere. Usate il seguente programma per leggerli:

```
10 FOR J = 0 TO 7
20 PRINT PEEK(32768 + J)
30 NEXT J
```

I numeri che si ottengono sono, in ordine: 28 (come prima), 34, 74, 86, 76, 32, 30 e 0. La figura 7.3 mostra come vengono interpretati. Nel generatore questi numeri compaiono come otto cifre binarie (o bit), quindi occorre convertirli in questa forma. Essi sono scritti a partire dal basso a sinistra di una tabella 8×8. In cima abbiamo le potenze di due, che mostrano il corrispondente valore decimale delle cifre binarie di ciascuna colonna. Fate passare la riga fino a che non trovate un numero che è uguale o maggiore a quello sulla sinistra. Nella prima riga, 128, 64 e 32 sono tutti più grandi di 28. Il successivo è 16, che è minore. Scrivete '1' nella casella che corrisponde a quella colonna e a quella riga. Ora sottraete 16 a 28, ottenendo 12. La prossima colonna porta scritto in testa '8', che è minore di 12, quindi scrivete ancora '1' in quella colonna. Sottraete 8 a 12, ottenendo 4. Sulla colonna successiva compare '4', quindi ponetevi un altro '1'. Se sottraete 4 a 4 il resto è zero, quindi la linea è finita. Nella seconda linea, dovete porre un '1' nelle colonne '32' e '2'. In ogni linea dovete scrivere un '1' nella colonna che porta scritto in alto un valore inferiore al numero a sinistra. Quando avete finito scrivete '0' in tutte le caselle vuote della tabella. Il risultato è un gruppo di cifre binarie che rappresentano un carattere. Ma quale? Per scoprirlo, considerate una seconda tabella e annerite i quadratini che corrispondono ad un '1' nella prima (fig. 7.4)

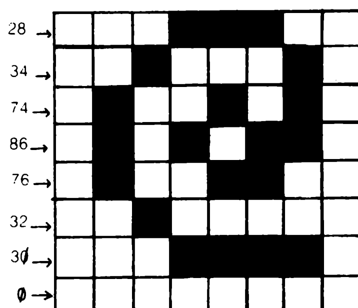


Fig. 7.4 - Un '1' nella Fig. 7.3 visualizza un punto sullo schermo, in modo da tracciare il carattere '@'.

Il risultato è l'immagine del simbolo '@'. Questo è il primo carattere del generatore. Potete vedere nella tabella 6 (Appendice A) che '@' ha come codice di carattere 0. Provate a leggere dalla memoria mediante l'istruzione PEEK la lettera A (dall'indirizzo 32776), usando lo stesso programma e ricostruendo 'A' sulla tabella.

Dall'esempio visto, potete rendervi conto che le istruzioni per visualizzare un carattere consistono in numeri di otto cifre binarie, memorizzate in otto indirizzi successivi del generatore di caratteri. In ogni numero la cifra '1' rappresenta un punto stampato sullo schermo. Uno '0' significa l'assenza del punto. Dopo aver scoperto ciò, tutto quello che dovete fare per definire un nuovo carattere è determinare i numeri binari che lo rappresentano. Ma attenzione! Il generatore di caratteri è simile alla ROM; la sua memoria non può essere cambiata, cioè non potete scrivere un gruppo di valori in qualsiasi sua parte. Naturalmente memorizzeremo i numeri binari su una RAM e faremo in modo che il VIC legga le istruzioni dalla RAM invece che dal generatore di caratteri.

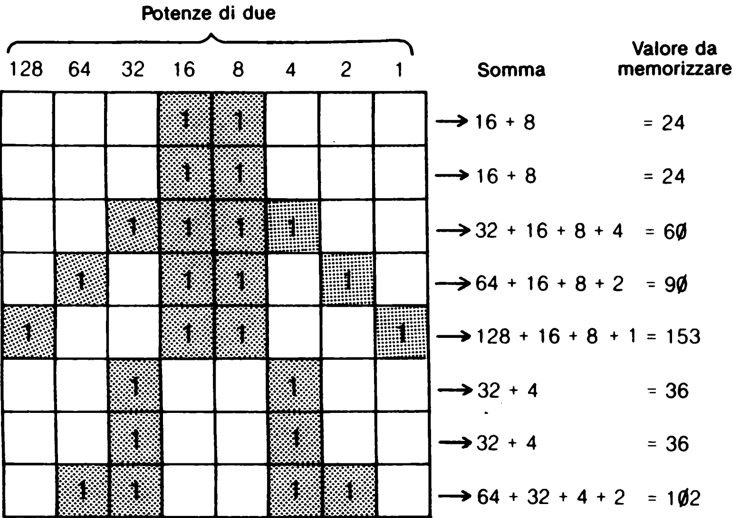


Fig. 7.5 - Calcolo dei valori necessari per la definizione di un nuovo carattere. (Nota: questa operazione è eseguita dal programma!).

Come creare un nuovo carattere

La figura 7.5 mostra il primo passo da fare. Il carattere è rappresentato mediante una tabella 8×8. Per ogni riga sommiamo i numeri che stanno in cima a quelle colonne che contengono un quadrato annerito. Questi

valori devono essere scritti mediante POKE in una parte adatta della memoria. È meglio usare gli indirizzi più alti, in modo che la nostra nuova 'RAM dei caratteri' e i programmi BASIC non interferiscano a vicenda. Il Listato 7.2 mostra un programma in cui un'istruzione DATA contiene i numeri che devono essere memorizzati.

```
10 FOR J = 0 TO 7
20 READ X
30 POKE 7168+J,X
40 NEXT J
50 POKE 36869,255
60 PRINT"□"
70 END
100 DATA 24,24,60,90,153,36,36,102
```

Listato 7.2 - Come visualizzare sullo schermo del VIC un nuovo carattere introdotto dalla tastiera.

Le linee dalla 10 alla 40 leggono mediante READ i dati e poi vengono scritti (POKE) in otto successive celle di memoria, a partire dalla 7168. La linea 50 altera il valore di uno dei registri del chip del VIC, in modo che il computer consideri come primo indirizzo del generatore di caratteri il 7168.

Quando lanciate il programma si cancella lo schermo e appaiono in alto a destra dei punti e dischi mischiati assieme. In questo modo il computer visualizza il solito messaggio 'READY'. Esso sta leggendo le istruzioni dalla RAM, e sta usando tutti gli '1' e '0' che vi trova. Per vedere che il programma funziona, premete il tasto '@'. Ogni volta che usate questo tasto, appare sullo schermo l'immagine di un omino. Il tasto '@' è stato impiegato per stampare un nuovo carattere.

La definizione di un nuovo carattere può essere un'operazione noiosa. Potete impiegare molto tempo per scrivere la tabella e sommare i numeri. Il programma Newkey nel Listato 7.3 vi permetterà di disegnare i caratteri sullo schermo. Vi fornisce i numeri che dovete memorizzare nella RAM e il relativo indirizzo. Seguendo il funzionamento di questo programma potrete capire meglio come lavora il Chip di Interfaccia Video. Non è possibile trattare in modo esauriente questo argomento, ma i cenni esposti sono una guida per chi vuole provare qualche esperimento. Se volete solamente usare il programma e non siete interessati a come funziona il chip, passate alla sezione 'Impiego di Newkey', dove sono elencate le istruzioni operative.

La linea 10 modifica i dati già introdotti negli indirizzi 51, 52, 55 e 56 in modo che il computer consideri l'inizio della RAM alla cella 32767, invece della sua usuale posizione. Ciò evita di memorizzare un programma o variabili stringhe nello stesso luogo riservato alle definizioni dei caratteri. Le linee 20 e 50 copiano il contenuto delle prime 1024 celle della ROM del generatore dei caratteri nella RAM a partire dall'indirizzo 6144.

```
10 POKE 51,255:POKE 52,23:POKE 55,255
20 POKE 56,23:FOR J=0 TO 1023
30 X=PEEK(32768+J):IFJ>255 AND J<264 THENX=255
40 POKE 6144 + J,X
50 NEXT J
60 POKE 36869,254:POKE 36866,136
65 POKE 36864,26:POKE 36867,144
70 PRINT"♥"
80 INPUT X$
90 IF X$ = "E" THEN GOTO 220
100 POKE 36879,157
110 FOR J = 0 TO 7
120 FOR K = 0 TO 7
130 P = 38400 + 8*J+K
140 P = PEEK(P) AND 7
145 IF P = 6 THEN B(J,N)= B(J,N) + 2↑(7-K)
150 NEXT K:NEXT J
160 PRINT"♥":INPUT K$(N)
170 FOR J = 0 TO 7
180 POKE 5632 + 8*ASC(K$(N)) + J,B(B(J,N)
190 NEXT J
200 POKE 36879,27:B(8,N)=5632+8*ASC(K$(N))
210 N=N+1:GOTO 70
220 POKE 36864,12:POKE 36866,150:POKE 36867,174
230 FOR J = 0 TO 7:POKE 6400+J,0:NEXT J
240 PRINT"♥"
250 PRINT K$(L);"  ";ASC(K$(L))
260 FOR J = 0 TO 8
270 PRINT B(J,L);" ";
280 NEXT J
290 GET Z$:IFZ$ = "" THEN GOTO 290
300 IF L = N - 1 THEN L = -1
310 L = L + 1:GOTO 240
```

Listato 7.3 - Newkey, un programma utile che vi aiuta a creare un nuovo carattere dalla tastiera.

In effetti stiamo scrivendo un nuovo carattere nella RAM del generatore. Non sono tutti i caratteri, ma solo le lettere maiuscole, i numeri, i simboli matematici, di punteggiatura e quelli grafici di destra. Lo 'spazio' (memorizzato nelle celle dalla 32324 alla 32331) non viene copiato come uno spazio, ma come un quadrato pieno. Come potete vedere nella linea 30, queste celle sono riempite con '255', che fornisce la soluzione 'tutti i punti' per ogni riga della matrice del carattere. Spiegheremo la ragione di ciò fra un attimo.

Il computer visualizza i caratteri in blu, come al solito, prima che il programma sia lanciato.

La linea 60 modifica il contenuto di diversi registri del Chip di Interfaccia. L'istruzione POKE agisce rispettivamente su: il registro 5, per cambiare la base degli indirizzi nella memoria riservata ai caratteri da 32768 a 6144; il registro 3, per dividere lo schermo in otto colonne invece che in ventidue; il registro 4, per ridurre la profondità dello schermo, da ventitre a otto righe, per visualizzare i nuovi caratteri.

Una istruzione INPUT alla linea 80 fa attendere il computer fino a che

non avete disegnato i nuovi caratteri.

L'utente preme SHIFT e CLR/HOME per spostare il cursore in alto a sinistra. Premendo la barra, visualizzate un gruppo di quadrati pieni di colore blu; ecco perchè abbiamo cambiato lo spazio in un quadrato. Premendo il tasto a sinistra CRSR, il cursore si muove senza stampare in blu. In questo modo l'area dello schermo viene divisa e la figura compare in bianco e blu. Quando l'immagine è completa, premete RETURN.

Lo schermo e il bordo cambiano colore. Ciò indica che sta accadendo qualcosa, ma non appare nulla sul video se non la nuova figura. Nelle linee dalla 110 alla 150, il computer legge la RAM dei codici dei colori dello schermo per trovare quello di ciascun quadrato. Questa memoria inizia dall'indirizzo 38400 (vedere fig. 3.3), ma dal momento che ci sono solo 64 celle, arriva fino a 38463. Viene letto il contenuto di ogni cella per conoscere i valori dei tre bit meno significativi (linea 140). L'operatore AND è usato in modo differente da come è stato spiegato nel capitolo sei (vedere fig.7.6). Il risultato è che P assume il valore del codice del colore (tabella 7 nell'Appendice A) per ogni quadrato sullo schermo. Se P = 6, è in blu, e ciò significa che deve essere stampato un punto come carattere finale. I numeri binari che servono sono raggruppati nel vettore B. Questo ha J colonne, corrispondenti alle otto righe dello schermo, e vi scriviamo

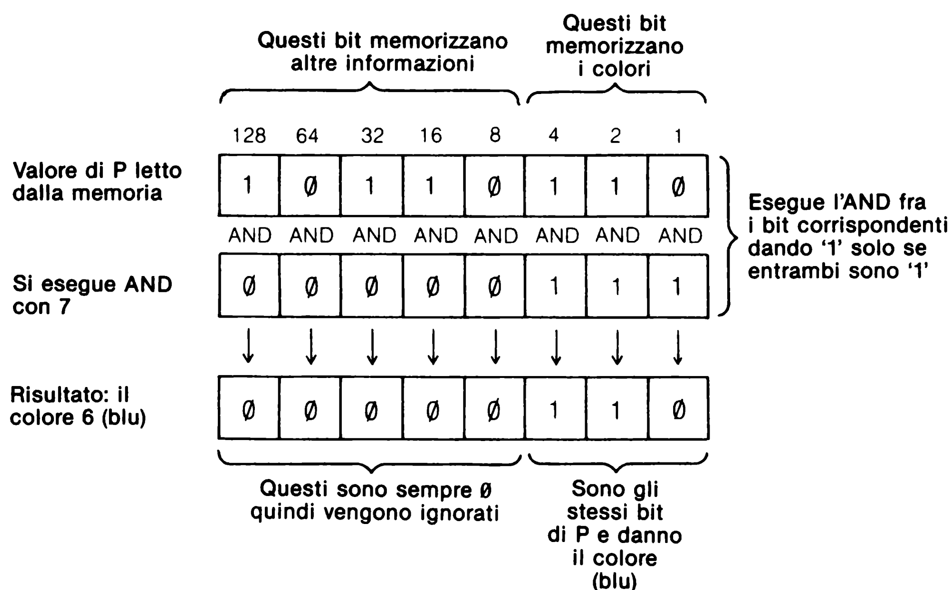


Fig. 7.6 - Spiegazione di come la linea 140 di Newkey determina il colore del quadrato.

un numero totale, non appena la linea è stampata. Il vettore B ha al massimo 11 colonne che ci consentono di tracciare 11 differenti caratteri in una volta sola. B non è dimensionato e pertanto N non può essere maggiore di 10.

L'espressione che segue il THEN nella linea 140 può ingannare, ma somma semplicemente le 'potenze di 2', al termine di ogni riga. Per esempio, per il quadrato di sinistra, K vale 0, e, se il colore è blu, B(N,J) è incrementato di 2 elevato a 7, cioè 128 (fate un confronto con la figura 7.3). Per il quadrato successivo, K vale 1. Se il suo colore è blu, B(N,J) è incrementato di 2 elevato a 6, cioè 64.

Alla linea 160, si chiede all'utente di premere un 'tasto'. Esso è il tasto corrispondente al carattere che deve essere memorizzato. Quando viene premuto, (seguito da RETURN), la macchina legge i valori dal vettore B e mediante POKE li scrive nelle celle di memoria della nuova RAM creata per i caratteri. Potete sapere l'indirizzo usando il codice ASCII del tasto che avete appena premuto (K$). Quest'ultimo viene fornito dalla funzione ASCII che appare nella linea 200. Poichè le celle di memoria riservate per i caratteri della tastiera sono in gruppo di otto, in accordo con il codice ASCII, gli indirizzi esatti si calcolano facilmente. Questi valori per il nuovo carattere che deve essere disegnato sono posti nella RAM, sovrapponendosi a quelli del carattere introdotto da tastiera precedentemente memorizzati nelle linee dalla 20 alla 50. Per questo motivo scegliete sempre caratteri come '@', '$' o '/' che non sono necessari.

Alla linea 210 si ritorna alla 70 per disegnare il carattere successivo. Se non volete alcuna altra figura, premete 'E' per uscire e il computer va alla linea 210 per stampare alcuni particolari.

Alla 210 l'istruzione POKE lavora su dei registri per ripristinare l'ampiezza del bordo, per allargare lo schermo a 22 colonne e a 23 righe. La linea 230 ridefinisce lo 'spazio' come uno 'spazio nullo' (tutti zeri).

Infine sono stampate le figure di tutti i caratteri.

Impiego di Newkey

(1) Scegliete come colore di stampa il blu (CTRL con 6).
(2) Lanciate il programma; impiega circa 25 secondi per trasferire i codici dei caratteri alla nuova area di RAM. Al termine di questa operazione lo schermo appare diviso in 8 linee per 8 colonne (e il bordo è colore ciano).
(3) Premete SHIFT assieme a CLR/HOME; il cursore si posiziona in alto a sinistra. Esso appare come una macchia, ma ciò non è rilevante.
(4) Per costruire il carattere, muovetevi con il cursore per tutto lo schermo, riga per riga, dalla prima all'ultima. Per disegnare un quadrato blu

premete la barra, mentre se lo volete bianco usate il tasto CRSR.
(5) Se avete commesso un errore, premete SHIFT e CLR/HOME e ripartite dal punto (3). Non usate INST/DEL.
(6) Alla fine dell'ultima riga, il cursore esce dallo schermo e non è più visibile. Se siete soddisfatti del disegno, premete RETURN, altrimenti iniziate di nuovo dal punto (3).
(7) Il bordo diventa verde, dimostrando che il codice è stato interpretato. Dopo di che appare ancora il cursore, sempre a forma di macchia.
(8) Decidete quale tasto volete allocare in quel carattere e premetelo (senza SHIFT). Vi sono dei tasti che non possono essere introdotti e che non vengono accettati; i due punti e la virgola. Sarebbe meglio evitare di usare i tasti dei numeri, poichè è più difficile leggerne i codici al passo (10).
(9) Ora premete RETURN. Il bordo diventa ciano e lo schermo bianco. Ritornate indietro al passo (3) e ripetete tutta la sequenza di operazioni per altri caratteri (10 al massimo). Quando avete finito, premete 'E' e poi RETURN.
(10) A questo punto ottenete la stampa finale (lo schermo ha di nuovo le consuete dimensioni). Per ogni carattere vengono visualizzati:

Il nuovo carattere.
Il codice ASCII del tasto usato per questo nuovo carattere (nel caso in cui ve lo foste dimenticato - il programma non può naturalmente visualizzare la lettera o il simbolo che appartiene a quel tasto).
Una lista di otto numeri, che sono quelli memorizzati nella RAM. Se volete usare i caratteri in un altro programma, copiate questi valori secondo l'ordine con cui appaiono.
Il primo indirizzo in cui sono stati scritti mediante l'istruzione POKE. Se necessario, copiate anche questo.

Premete ogni tasto per vedere i vari caratteri e i loro dettagli. Quando avete terminato, avete una tastiera in cui alcuni tasti hanno il loro usuale significato, mentre altri corrispondono ai caratteri che avete creato. Potete usarli per introdurre dati o per disegnare grafici sullo schermo. Possono essere impiegati nei programmmi di giochi o di altro tipo. Se volete ritornare alla tastiera normale, battete la seguente linea:

```
POKE 36869,240
```

Quando premete RETURN, tutti i nuovi caratteri che comparivano sullo schermo sono convertiti istantaneamente in quelli corrispondenti ai tasti ai quali erano stati assegnati. La tastiera si comporta normalmente, e l'introduzione dei programmi risulta più semplice. Tuttavia i nuovi ca-

ratteri sono ancora presenti (a meno che non spegnete il computer o premete RUN/STOP e RESTORE) e potete riutilizzarli battendo:

POKE 36869,254

Se volete scrivere un programma per utilizzare questi caratteri, cancellate Newkey. Troverete che avete a disposizione 2K di memoria. In qualsiasi parte del programma o anche in esecuzione potete passare dai nuovi ai vecchi caratteri e viceversa, usando uno dei due comandi POKE sopra esposti. Quando state scrivendo un programma nel modo normale, usate i tasti speciali che avete definito quando volete far apparire i nuovi caratteri. Quando state programmando essi compaiono nella loro veste usuale, ma quando mediante POKE ritornate al nuovo gruppo di simboli, riappariranno i caratteri che avete disegnato.

Più avanti in questo capitolo useremo ancora questo procedimento. A proposito, non usate il modo di testo o i caratteri in reverse, perchè essi non sono presenti nella vostra RAM.

Multikey

I caratteri normali del VIC e quelli definiti dal programma Newkey sono in due colori:

(1) Il colore del primo piano (a volte indicato come colore di 'stampa'), che di solito è blu.
(2) Il colore dello sfondo (a volte detto colore dello schermo), che generalmente è bianco.

Usando uno speciale modo grafico del VIC chiamato Modo Multicolore, è possibile aggiungere due altri colori ai caratteri grafici:

(3) Il colore del bordo, cioè lo stesso colore del bordo dello schermo televisivo.
(4) Un colore ausiliario, cioè un qualsiasi altro quarto colore.

Il colore di primo piano e del bordo possono essere scelti tra gli otto elencati in cima alla Tabella 5 dell'Appendice A. Quello di sfondo e il quarto tra i 16 a sinistra in basso sempre della stessa Tabella.

Come vi potete aspettare, l'uso di colori aggiuntivi comporta una perdita in qualche altra direzione. In ogni riga di caratteri i punti sono in quattro coppie e i membri di ciascuna coppia debbono essere dello stesso colore. In effetti ogni linea consiste in quattro coppie di punti. La risoluzione che si ottiene in una grafica multicolore non è così precisa come quella

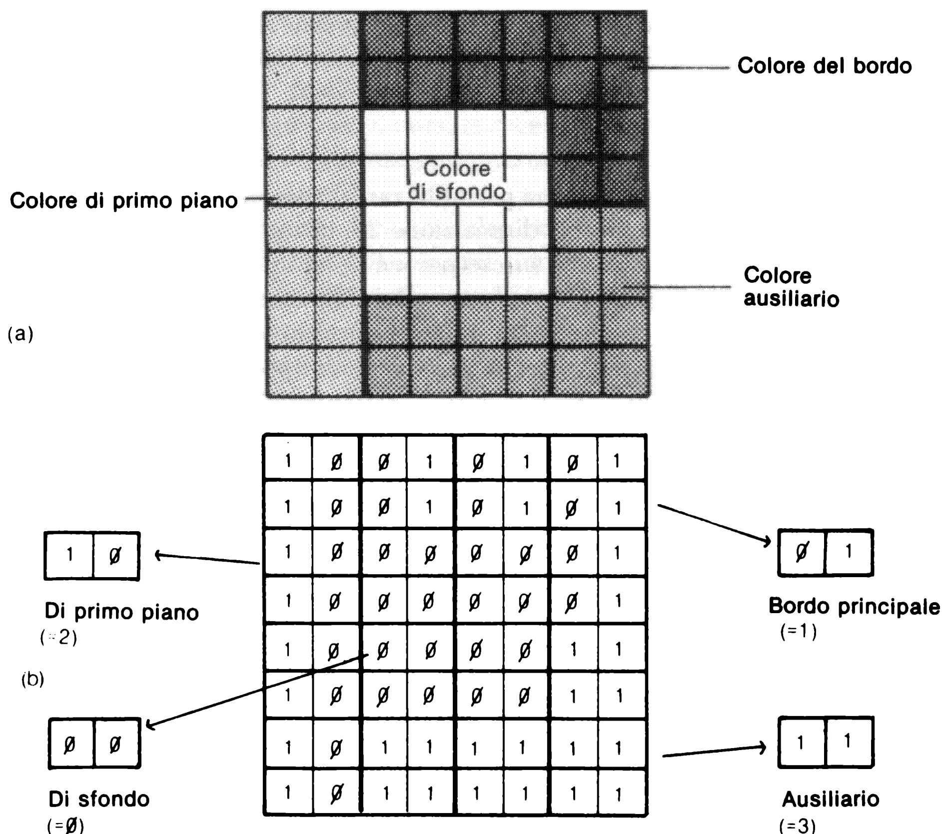


Fig. 7.7 - Il disegno di un carattere multicolore. (a) Visualizzazione delle aree colorate. (b) La configurazione di bit corrispondente; in ogni riga i bit sono determinati in coppie.

a due colori usuale, per lo meno se visualizzata su di uno schermo televisivo. A meno che non abbiate un monitor a colori, non realizzate disegni troppo complicati.

I dati per un carattere multicolore sono memorizzati in un numero binario di 8 bit, come per i caratteri a due colori. La differenza è che per rappresentare i colori nel secondo caso bastano un '1' e uno '0', nel primo occorrono due bit. I codici dei quattro colori sono:

00 colore di sfondo
01 colore del bordo
10 colore di primo piano
11 colore ausiliario

I bit di ciascun numero binario sono letti due alla volta e interpretati co-

```
10 POKE51,255:POKE52,23:POKE55,255:POKE56,23
20 FOR J=0 TO 1023
30 X=PEEK(32768+J):IFJ>255ANDJ<264 THENX=255
40 POKE 6144 + J,X
50 NEXT J
51 PRINT"□"
52 INPUT"BORDO";BO
53 INPUT"SFONDO";BA
54 BO=BO - 1:BA=BA-1:IF BA>8 THEN BA=BA-2
56 INPUT"PRIMO PIANO";FO
57 INPUT"AUSILIARIO";AU
58 FO=FO-1:AU=AU-1:IF AU>8 THEN AU=AU-2
60 POKE 36869,254:POKE 36866,136
65 POKE 36864,26:POKE 36867,144
70 PRINT"□"
80 INPUT X$
90 IF X$="E" THEN GOTO 220
100 POKE 36879,157
110 FOR J = 0 TO 7
120 FOR K = 0 TO 7
130 P = 38400 + 8*J+K
140 P = PEEK(P) AND 7
142 IF P=BO THEN B(J,N)=B(J,N)+4↑(3-K/2)
144 IF P=FO THEN B(J,N)=B(J,N)+2*4↑(3-K/2)
146 IF P=AU THEN B(J,N)=B(J,N)+ 3*4↑(3-K/2)
147 IF P=AU-8 THEN B(J,N)=B(J,N)+3*4↑(3-K/2)
150 NEXT K:NEXT J
160 PRINT"□":INPUT K$(N)
170 FOR J = 0 TO 7
180 POKE 5632+8*ASC(K$(N))+J,B(J,N)
190 NEXT J
200 POKE 36879,27:B(8,N)=5632+8*ASC(K$(N))
210 N=N+1:GOTO 70
220 POKE 36864,12:POKE 36866,150
221 POKE 36867,174:POKE 36878,16*AU
222 IF FO = 0 THEN PRINT"■"
223 IF FO = 1 THEN PRINT"▣"
224 IF FO = 2 THEN PRINT"▩"
225 IF FO = 3 THEN PRINT"◣"
226 IF FO = 4 THEN PRINT"▦"
227 IF FO = 5 THEN PRINT"▧"
228 IF FO = 6 THEN PRINT"▤"
229 IF FO = 7 THEN PRINT"▥"
230 POKE 36879,8 + 16*BA + BO
231 FOR J = 0 TO 7:POKE 6400+J,0:NEXT J
240 PRINT"□"
250 PRINT K$(L);"  ";ASC(K$(L))
255 POKE 38422,FO+8
260 FOR J = 0 TO 8
265 POKE 38422,FO+8
270 PRINT B(J,L);" ";
280 NEXT J
290 GET Z$:IF Z$="" THEN GOTO 290
300 IF L = N-1 THEN L=-1
310 L=L+1:GOTO 240
```

Listato 7.4 - Multikey, un programma utile che vi aiuta a disegnare caratteri grafici a più colori (4).

me spiegato. La fig 7.7 mostra come ciò è possibile. Per esempio, per ottenere una striscia del colore del primo piano lungo la parte sinistra del carattere, i primi due bit di ogni numero di codice devono essere '10'.

Definire un carattere significa perciò stabilire quali aree devono assumere i quattro colori disponibili. Poi dobbiamo agire mediante l'istruzione POKE sul Chip di Interfaccia Video per scegliere i colori dello schermo e del bordo, come abbiamo spiegato nel capitolo tre. Il colore di primo piano è determinato usando CTRL con un tasto numerico, mentre quello ausiliario ponendo il suo codice in un altro registro del VIC. Sfortunatamente tutti i caratteri multicolore visualizzati contemporaneamente sullo schermo devono fare uso degli stessi quattro colori. Essi possono essere usati assieme a quelli ordinari o a quelli che avete definito mediante Newkey.

Il programma Multikey (Listato 7.4) vi risparmia la fatica di definire caratteri multicolore. È assai simile a quello Newkey (Listato 7.3). Se siete più interessati all'uso piuttosto che alla comprensione del programma passate alla sezione 'L'uso di Multikey' per le istruzioni che dovete eseguire. La descrizione che faremo non è così dettagliata come per il programma Newkey. Questa volta le linee non sono numerate di dieci in dieci. La ragione è che alcune di esse sono le stesse del programma Newkey. Se lo avete introdotto precedentemente non impiegherete molto a modificare alcune linee e ad aggiungerne di nuove per realizzare il programma Multikey.

Le linee 51 e 57 chiedono all'utente di introdurre i colori desiderati. Dovete rispondere battendo uno dei tasti numerici da 1 a 8 (vedere le istruzioni alla sezione successiva). Le variabili BO, BA, FO, e AU sono usate per i codici dei colori.

Alla linea 80 il computer attende che l'utente tracci il suo disegno. Questa volta la barra serve per introdurre i quadrati in uno dei quattro colori, mentre il tasto CRSR non è usato. Il colore dei quadrati è definito premendo CTRL e un tasto numerico da 1 a 8.

La linea 120 esamina ogni linea per scoprire i colori. Entrambi i componenti di una coppia di quadrati devono essere dello stesso colore, e anche se ciò non si verifica, la cosa viene ignorata. Il colore di ogni elemento del disegno è determinato nella linea 140 e una delle linee dalla 142 alla 146 somma il giusto valore a quello memorizzato in B(J,N).

Se il quadrato ha il colore di sfondo (codice 00), non viene sommato nulla. Se ha il colore del bordo (codice 01) il valore sommato è una potenza di 4, secondo la posizione del quadrato (vedere fig. 7.8). Se il colore è quello di primo piano (codice 10, corrispondente a 2 in decimale), il valore aggiunto è due volte la potenza di 4. Se il colore è quello ausiliario (codice 11, cioè 3 in decimale), si somma la potenza di 4 moltiplicata per 3.

Potete definire parecchi caratteri, come nel programma Newkey, e alla

Potenze di quattro: 64	16	4	1	Somma	Valore da memorizzare
1 Ø = 2	Ø 1 = 1	Ø 1 = 1	Ø 1 = 1	→ 2×64+16+4+1	=149
1 Ø = 2	Ø 1 = 1	Ø 1 = 1	Ø 1 = 1	→ 2×64+16+4+1	=149
1 Ø = 2	Ø Ø = Ø	Ø Ø = Ø	Ø 1 = 1	→ 2×64+Ø+Ø+1	=129
1 Ø = 2	Ø Ø = Ø	Ø Ø = Ø	Ø 1 = 1	→ 2×64+Ø+Ø+1	=129
1 Ø = 2	Ø Ø = Ø	Ø Ø = Ø	1 1 = 3	→ 2×64+Ø+Ø+3×1	=131
1 Ø = 2	Ø Ø = Ø	Ø Ø = Ø	1 1 = 3	→ 2×64+Ø+Ø+3×1	=131
1 Ø = 2	1 1 = 3	1 1 = 3	1 1 = 3	→ 2×64+3×16+3×4+3×1	=191
1 Ø = 2	1 1 = 3	1 1 = 3	1 1 = 3	→ 2×64+3×16+3×4+3×1	=191

Fig. 7.8 - Calcolo dei valori necessari per la definizione del carattere multicolore della fig. 7.7. È realizzato per voi dal programma Multikey.

fine premete 'E'. Allora alla linea 120, l'ultimo elemento della linea pone il colore ausiliario nel registro 15 del Chip di Interfaccia. Abbiamo già usato questo registro per il controllo del volume nei generatori di suoni. I 4 bit meno significativi (fig. 7.9) controllano il volume, mentre i primi 4 servono per memorizzare il colore ausiliario. Per scrivervi il valore di AU, moltiplichiamolo per 16, come è indicato nella linea 220.

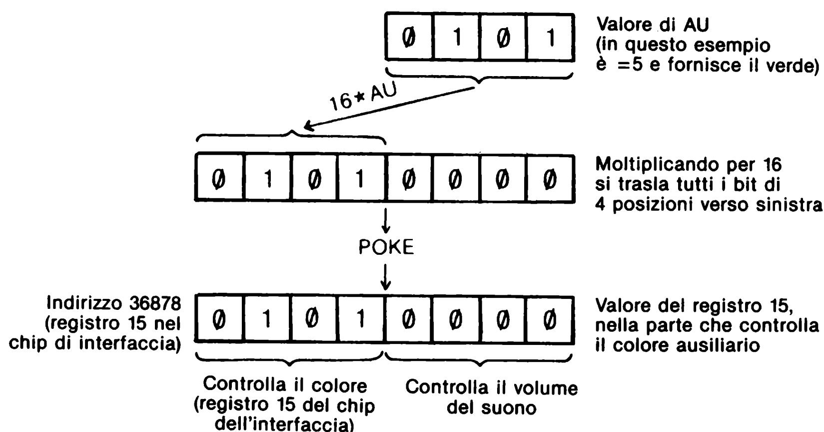


Fig. 7.9 - Controllo del colore ausiliario (linea 220 del Multikey). AU può assumere qualsiasi valore compreso tra 0 (0000) e 15 (1111).

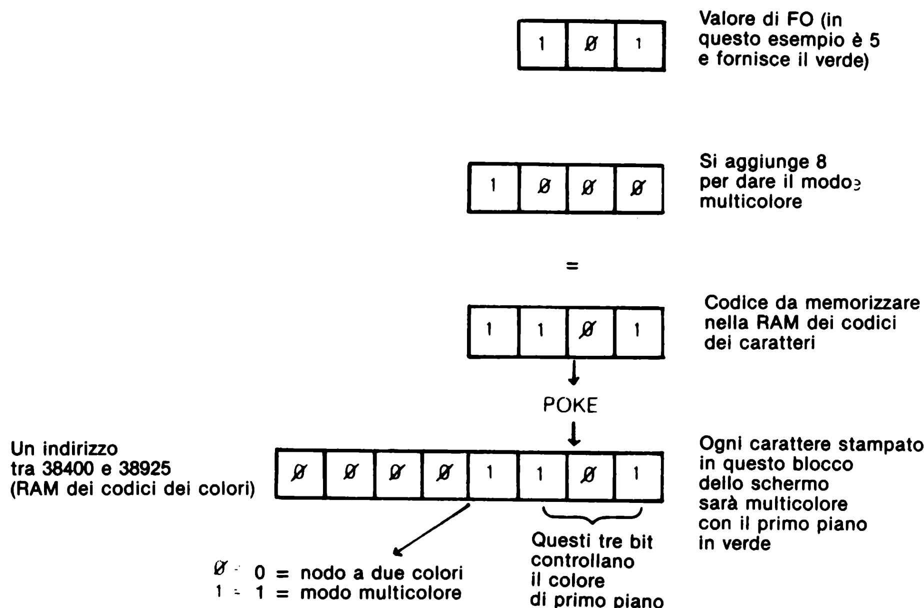


Fig. 7.10 - Controllo del colore di primo piano nel modo multicolore (linea 265 del Multikey). FO può assumere qualsiasi valore compreso tra 0 (000) e 7 (111).

Le 221 e 229 definiscono il colore di primo piano, la 229 quello di sfondo e del bordo. La routine che serve per visualizzare i risultato sullo schermo è la stessa del programma Newkey. L'unica differenza consiste nel fatto che dovendo stampare un carattere multicolore occorre aggiungere 8 (mediante POKE) al contenuto dei corrispondenti indirizzi nella memoria dei codici dei colori (fig. 3.3). Il carattere è stampato sulla seconda linea e alla seconda colonna, ed ha come indirizzo 38422. Scriviamo in quest'indirizzo il colore di primo piano sommato a 8. La figura 7.10 mostra cosa accade. Se non inizializzate l'indirizzo del codice del colore in questo modo, il carattere appare in due colori, e ciò può sembrare abbastanza strano!

L'uso di Multikey

(1) Non importa quale sia il colore dello schermo (di primo piano) con il quale inizia il programma.
(2) Lanciate il programma. Dopo 25 secondi vi si chiede di introdurre:

Il colore del bordo: tasto dall'1 all'8, secondo i colori segnati su di essi.
Colore di sfondo: tasto dall'1 all'8 o tasto 1 (per indicare che volete un

colore chiaro), seguito dal numero del colore voluto.
Colore di primo piano: tasto dall'1 all'8.
Colore ausiliario: tasto dall'1 all'8, o dall'11 al 18, come per il colore di primo piano.

Scrivete questi colori in modo che possiate poi ricordarveli.
Notate che con questo programma non è possibile avere un colore e la sua sfumatura più chiara nello stesso carattere. Ora appare lo schermo di 8 linee e 8 colonne.
(3) Premete SHIFT e CLR/HOME; quindi il cursore si posiziona in alto a sinistra dello schermo.
(4) Per costruire il carattere fate passare lo schermo riga dopo riga, dalla prima all'ultima. Per fare ciò usate la barra, non il tasto CRSR. Premete CTRL assieme a un numero tra 1 e 8, per scegliere il colore che deve avere ogni quadrato. Ricordate che i quadrati della stessa coppia devono essere del medesimo colore. Se fate un errore premete SHIFT e CRL/HOME e ripartite dal punto (3).
(5) Alla fine dell'ultima linea il cursore scompare dallo schermo. Se siete soddisfatti del disegno, premete RETURN. In caso contrario, cancellate lo schermo e iniziate di nuovo dal passo (3).
(6) Il colore del bordo diventa verde, dimostrando che il codice è stato interpretato. In seguito compare il cursore.
(7) Decidete quale tasto debba essere riservato per il carattere e premetelo (non usate i due punti e la virgola). Poi premete RETURN.
(8) Il bordo diventa ciano e ritornate al punto (3). Ripetete le operazioni fino ad introdurre al massimo 10 caratteri, se lo desiderate. Quando avete finito, battete 'E' e RETURN.
(9) Ora siete arrivati alla visualizzazione sullo schermo. Per ogni carattere potete vedere:

Il nuovo carattere
Il codice ASCII
Gli 8 codici
Il primo indirizzo.

L'interpretazione di questi dati è spiegata a pagina 94. Premete un qualsiasi tasto per vedere il carattere successivo.

L'uso dei caratteri

Dopo aver definito i caratteri e aver loro attribuito dei corrispondenti tasti, potete usare la tastiera per farli comparire sullo schermo o per incor-

porarli in istruzioni PRINT. Se non volete definire altri nuovi caratteri, cancellate il programma. Così facendo avete a disposizione 2K di memoria per immagazzinare i programmi.

A questo punto potete scriverne basandovi sui caratteri che avete definito. Nel capitolo undici ci sono alcuni esempi di questo tipo.

Quando spegnete il sistema o premete RUN/STOP e RESTORE, i nuovi caratteri vanno perduti. Se avete provveduto a ricopiare i loro codici alla fine dei programmi Newkey e Multikey, potete sempre recuperarli. Potete inserirli in qualsiasi altro programma che scriverete successivamente in modo da definire alcuni tasti speciali. I caratteri che avete definito in differenti occasioni possono essere usati assieme nello stesso programma. Alcuni possono essere stati creati mediante Newkey, altri da Multikey. Tutte le definizioni dei caratteri che preferite possono costituire una ‘biblioteca’.

Se siete interessati a programmi di giochi, la vostra collezione sarà formata da disegni di aereoplani, astronavi, mostri o simili, o forse dalle figure degli scacchi. Se volete impiegare il VIC in scopi educativi, troverete assai semplice disegnare i caratteri ‘$_2$’, ‘$_4$’ e gli altri usati nelle formule chimiche, come in H_2SO_4. Per la matematica potete scrivere gli esponenti, come ‘^{2}’ o ‘^{3}’ o altri simboli aritmetici, come Σ, $^\circ$ e $\approx$.

In alcuni programmi potreste voler usare sia i caratteri definiti da voi, sia quelli usuali, come le lettere, i numeri, i simboli e quelli della grafica della tastiera del VIC. Il modo per incorporare i due gruppi di caratteri sarà descritto nel capitolo undici.

Riassunto

In questo capitolo avete imparato come:

— usare il comando PRINT per visualizzare dei grafici
— usare il comando TAB per porre sullo schermo le stringhe nelle posizioni desiderate
— usare la funzione ASCII
— usare la funzione CHR$
— ridefinire i tasti per creare caratteri a due colori
— ridefinire i tasti per creare caratteri a più colori.

Capitolo 8
Operazioni sulle stringhe

Ora osserveremo un po' meglio le operazioni che potete realizzare con le stringhe. Avete già visto che le stringhe possono essere addizionate tra loro. Non si tratta di addizione in senso aritmetico, in quanto tutto quello che viene fatto consiste nel prendere due o più stringhe ed unirle l'una in coda all'altra. È possibile anche sottrarre stringhe tra loro? Non precisamente, ma esistono modi per estrarre parti di stringhe che possono essere considerati equivalenti di una sottrazione. Provate a battere queste istruzioni:

```
10 INPUT A$
20 INPUT N
30 B$=LEFT$(A$,N)
40 PRINT B$
```

Al primo input, introducete una parola, per esempio 'BUONASERA'. Al secondo input, battete un numero, preferibilmente minore del numero di lettere contenuto, nella parola. Se per esempio battete '5', la linea 30 fa sì che B$ sia formato dalle prime cinque lettere di A$. Sullo schermo leggerete 'BUONA'. LEFT$ è una funzione di stringa. Richiede due argomenti (così chiamiamo le voci contenute tra le parentesi dopo la parola LEFT$). Il primo argomento è il nome della stringa su cui si deve lavorare. Il secondo indica il numero di lettere che devono essere estratte da quella stringa, cominciando a contare da sinistra. Lanciate il programma alcune volte finchè non avete asssimilato l'idea di ciò che è LEFT.

Nel programma precedente B$ è semplicemente uno stadio intermedio, tant'è che le linee 30 e 40 possono essere sostituite da

```
30 PRINT LEFT$(A$,N)
```

RIGHT$ ha una funzione analoga, operando però sulla stringa a partire da destra

```
$ 10 INPUT A$
20 INPUT N
30 PRINT RIGHT (A$,N)
```

Se introducete 'BUONASERA' e poi '4', leggerete sullo schermo 'SERA'.

Il terzo membro di questo trio di funzioni di stringa è MID$. Come vi potete aspettare dal nome, questo comando preleva dalla stringa una fetta di essa (sotto-stringa). Questa operazione richiede un ulteriore valore negli argomenti. Il primo argomento, come prima, indica la stringa. Il secondo la posizione all'interno della stringa dalla quale far cominciare la sotto-stringa. Il terzo rappresenta il numero di lettere che sono contenute nella sotto-stringa. Per esempio, se A$ è 'BUONASERA', allora l'istruzione MID$(A$,4,4) fa apparire sul video 'NASE'.

Il programma di visualizzazione dell'ora nel listato 8.1 mostra un altro utilizzo di queste funzioni di stringa.

```
10 INPUT"ORE";H$
20 INPUT"MINUTI";M$
30 TI$=H$ + M$ + "00"
40 IF RIGHT$(TI$,2) <> "00" THEN GOTO 40
50 PRINT"♥"
60 H$ = LEFT$(TI$,2)
70 M$ = MID$(TI$,3,2)
80 PRINT"SONO LE":PRINT
90 PRINTTAB(8)H$;".";M$
100 FOR J = 1 TO 1000: NEXT J
110 GOTO 40
```

Listato 8.1 - Il programma dell'orologio mostra alcuni dei possibili utilizzi delle stringhe.

Le ore ed i minuti sono introdotti come stringhe, H$ e M$. Per queste due deve essere predisposto un formato di due cifre ciascuna. Per esempio se sono le due e cinque, le ore sono '02' ed i minuti '05'. Potete migliorare il programma scrivendo sottoprogrammi che permettano l'uso di cifre singole.

La linea 30 forma una stringa di sei cifre come richiesto da TI$. I secondi non vengono introdotti e così le ultime due cifre sono '00'. La linea 40 esamina ripetitivamente TI$ per vedere se le ultime due cifre sono '00'. Questo succede tutte le volte che l'ora ha un numero esatto di minuti dall'istante iniziale. RIGHT$ mostra qui la sua utilità.

Il programma attende alla linea 40 finchè non passa esattamente un minuto, poi la macchina passa dalla linea 50 fino alla 90 per visualizzare l'ora. Le ore ed i minuti sono estratti da TI$ utilizzando LEFT$ e MID$. Abbiamo bisogno del terzo e del quarto carattere della stringa TI$ per le ore, così prendiamo due caratteri, a partire dal terzo. La linea 100 inserisce un piccolo ritardo nel programma perchè se la macchina tornasse immediatamente alla linea 40 le ultime due cifre di TI$ potrebbero essere

ancora '00' e verrebbe prodotto un segnale sfarfallante.

Questo programma mostra alcune altre funzioni che operano sulle variabili di stringa. Così com'è visualizza l'ora in base al suo orologio impostato sulle 24 ore: potrebbe essere cambiato e reimpostato sulle 12 ore. Quello che dovete fare è inserire un test su H$ per vedere se contiene un valore maggiore di 12. Se sì, sottraete 12 da questo. Le variabili di stringa non possono essere trattate come se fossero delle variabili numeriche, anche se come in questo programma, contengono numeri. Esiste la funzione VAL che converte una stringa in una variabile numerica. C'è anche la funzione inversa STR$. La linea da aggiungere al programma di visualizzazione dell'ora le utilizza entrambe:

```
65 IF VAL(H$)>12 THEN H$=STR$(VAL(H$)—12)
```

Il VAL(H$) che segue la IF converte la H$ in un numero. Essa può ora essere confrontata con il numero 12. Se è maggiore, allora H$ deve essere cambiato. Ecco come: la funzione STR$ ha come argomento (VAL(H$)—12). Se per esempio il valore di H$ fosse 15, verrebbe ridotto di 12, dando 3. Poi STR$ converte 3 nella stringa '3'. H$ viene posto uguale a questa nuova stringa. Il risultato che ottenete è che il programma visualizza l'ora come 3.23 piuttosto che 15.23.

VAL e STR$ sono una potente coppia di comandi per lavorare con stringhe che contengono numeri. Se VAL viene applicato ad una stringa che contiene lettere e numeri misti tra loro, legge la stringa da sinistra verso destra fermandosi al primo carattere che non sia un numero. Dunque:

Se A$='23 High Street', VAL(A$) dà 23

Se A$='BUONASERA', VAL(A$) dà 0 in quanto il primo carattere non è un numero.

Se A$='1 APRILE 1984', VAL$ dà 1 e 1984 dopo il nome del mese è ignorato.$

Tre caratteri non numerici che vengono accettati da VAL sono '+', '—', '.' (punto decimale).

Se avete aggiunto al programma la linea 65, avrete notato che le ore vengono talvolta visualizzate precedute da uno zero (ad esempio 02.35), talvolta senza (2.35). Nelle prime ore del giorno, e cioè quando VAL(H$) è minore di 12, la linea 65 non interviene. H$ è una stringa a due caratteri ed inizia per 0 in quanto è stata ottenuta estraendo due caratteri da TI$. In seguito, dopo il mezzogiorno, la linea 65 agisce e (VAL(H$)—12) genera un numero inferiore a 12. Questo non viene preceduto da zero in quanto è un numero e non una stringa. Poi STR$ lo converte in stringa così com'è, senza essere preceduto dallo zero.

Se preferite togliere lo zero, questo può essere facilmente realizzato ope-

rando su H$ con RIGHT$:

63 IF VAL(H$)<10 THEN H$=RIGHT(H$,1)

Prima di abbandonare il programma di visualizzazione dell'ora potreste sbizzarrirvi a migliorarlo in più modi. È semplice includervi la possibilità di utilizzo come sveglia. All'utente viene richiesto di introdurre l'ora della sveglia. Questo potrebbe essere memorizzato come A$, una stringa composta da sei cifre da confrontare con TI. Quando TI$=A$ il, programma passa ad una subroutine che, per esempio, fa lampeggiare i colori dello schermo, o che avvia qualche segnale acustico. Altra modifica realizzabile è fare stampare A.M. o P.M. secondo se siete prima o dopo il mezzogiorno. Una modifica più sofisticata del programma potrebbe consistere nel fare rintoccare le ore, le mezze ed eventualmente anche i quarti d'ora.

Longspell

Questo è un gioco di compitazione che ci mostra l'uso di un'altra funzione di stringa. Viene utilizzato per calcolare il numero di caratteri contenuti in una stringa. Se A$='ANYTHING', per esempio, LEN(A$)=8. Come numero di caratteri si contano anche numeri, punteggiatura e spazi, così che se per esempio A$=' HALF-PAST 2 ', con uno spazio all'inizio

```
10 FOR J = 1 TO 6
20 N = INT(RND(1)*90)
30 IF N < 65 THEN GOTO 20
40 A$(J) = CHR$(N)
50 NEXT J
60 PRINT"□":FF = 0
70 FOR J = 1 TO 6
80 PRINTA$(J);"   ";
90 NEXT J
100 PRINT:PRINT"QUAL'E' LA VOSTRA PAROLA?"
110 PRINT:INPUTW$
120 FOR J = 1 TO LEN(W$)
130 F = 0
140 T$ = MID$(W$,J,1)
150 FOR K = 1 TO 6
160 IF A$(K)=T$ THEN F = 1
170 NEXT K
180 IF F=0 THEN FF=1
190 NEXT J
200 IF FF=1 THENPRINT"CONTIENE ALTRE":GOTO230
210 PRINT:PRINT LEN(W$);"LETTERE"
220 IF LEN(W$)>M THEN M=LEN(W$):PRINT
225 PRINT"LA PAROLA PIU' LUNGA FINORA E'"
230 GET X$:IF X$="" THEN GOTO 230
240 IF X$="N" THEN GOTO 10
250 GOTO 60
```

Listato 8.2 - Longspell, un gioco semplice che stimola a pensare.

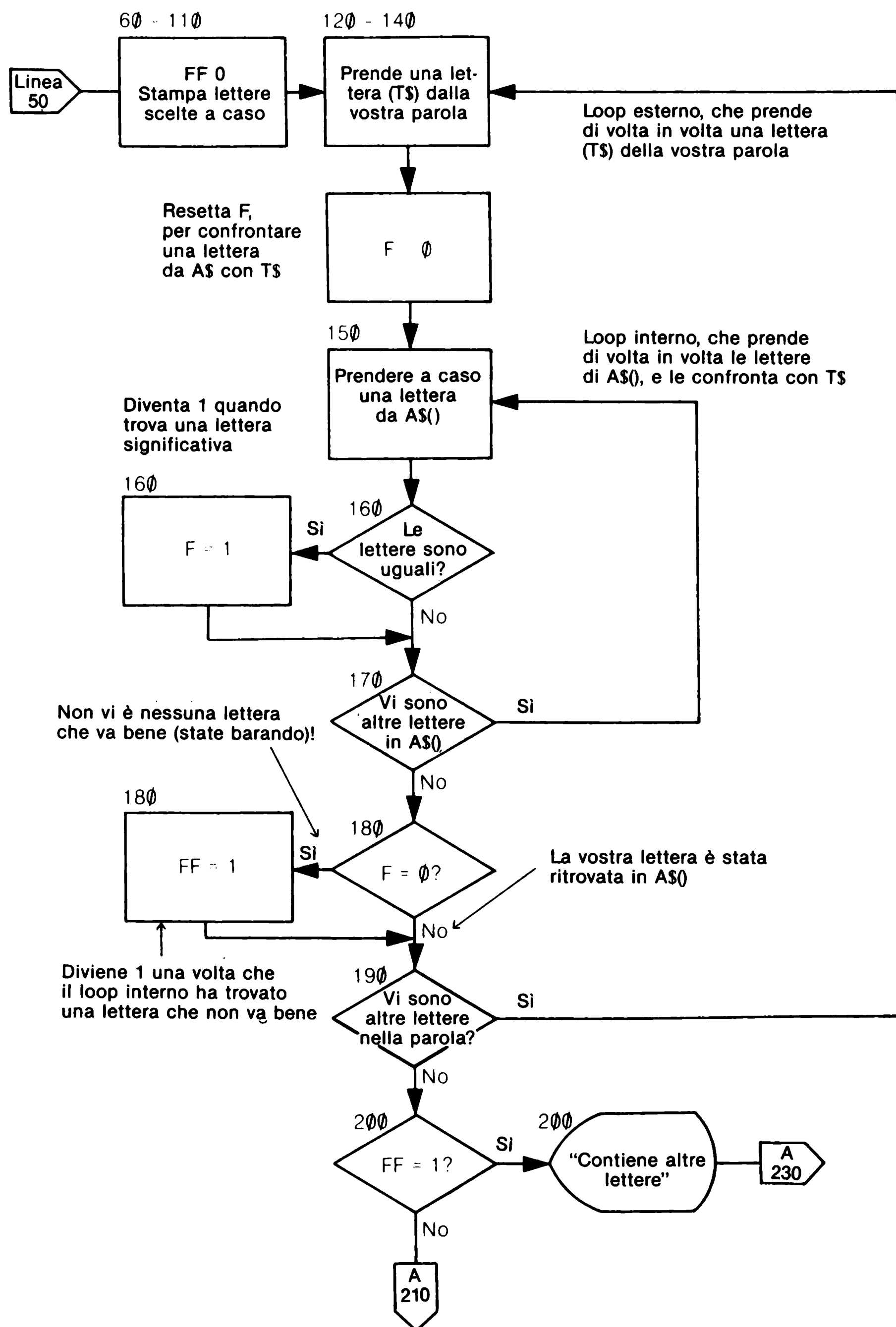


Fig. 8.1 - Il controllo della vostra parola in Longspell.

ed uno alla fine, LEN(A$)=13.

'Longspell'(vedere il listato 8.2) comincia con una routine che sceglie a caso sei lettere dell'alfabeto. La linea 20 sceglie un numero a caso, compreso tra 1 e 89, mentre la linea 30 scarta tutti i numeri al di sotto di 65. Questa operazione ci lascia tutti i numeri tra 65 e 89. Se guardate la tabella 10 dell'appendice A, vedrete che i numeri in questo intervallo sono i codici ASCII per le lettere dell'alfabeto da A ad Y. Utilizzando CHR$, il numero casuale genera casualmente una lettera da mettere nel vettore A$, che alla fine conterrà sei lettere così trovate.

Non è determinante trascurare la Z, in quanto lo scopo del gioco è scrivere le parole più lunghe possibili, usando le sei lettere scelte a caso. Le stesse lettere possono essere ripetute quante volte si vuole all'interno della parola. Il gioco può essere fatto da due o più giocatori, per vedere chi sa ottenere la parola più lunga dalle lettere assegnategli. Se giocate da soli, potete provare a battere il vostro record personale sulla parola più lunga. Una variazione consiste nello scrivere la frase più lunga utilizzando le lettere assegnate.

Alla linea 110 introducete la vostra parola. Ora il computer controlla la vostra parola (W$) lettera per lettera, per vedere che voi non abbiate barato usando lettere che non erano state scelte. Il controllo viene realizzato con un loop tipo FOR...NEXT. LEN(W$) nella linea 120 calcola la lunghezza della vostra parola, così il computer sa quante lettere deve controllare. Ogni volta, nel loop, preleva una lettera dalla vostra parola, usando MID$ (linea 140) e la battezza T$. Deve poi confrontare T$ con ogni lettera nel vettore A$; deve essere uguale ad una di queste. Occorre che faccia ciò per ogni lettera della vostra parola e che si accerti di volta in volta che la prima lettera della vostra parola E la seconda E la terza E così via vadano bene. Tutta l'operazione avrebbe potuto essere scritta con operazioni logiche, con un numero di OR ed AND tale da non stare nemmeno su di una riga. Due flag, F e FF, sono usati per tener conto della logica. Il loro funzionamento è spiegato in fig. 8.1.

La linea 200 rigetta la vostra parola se contiene una lettera che non è nel vettore A$. Alla linea 210, LEN(W$) viene usato per trovare quante lettere contiene la vostra parola. La linea 220 mostra come tenere aggiornato un record del valore massimo di una serie di numeri. Il numero massimo è indicato con M, che vale zero all'inizio del programma. Ogniqualvolta che LEN(A$) supera M, M assume questo nuovo valore. In questo modo M cresce tutte le volte che voi componete parole più lunghe. La linea 230 aspetta che voi leggiate ciò che vi viene visualizzato. Se volete riprovare a trovare nuove parole usando lo stesso gruppo di lettere, premete qualsiasi tasto tranne 'N'. Premete 'N' per far scegliere alla macchina un nuovo gruppo.

Il bene è più grande del male?

Allo stadio di sviluppo attuale, i computers non sono in grado di risolvere questioni filosofiche del tipo proposto nel titolo del paragrafo. Possiamo ovviamente usare LEN per chedere se la parola 'bene' è più grande della parola 'male', ma è cosa malfatta in quanto la risposta è ovvia a priori. Vediamo come ci risponde la macchina realizzando questo programma:

```
10 INPUT A$
20 INPUT B$
30 IF A$<B$ THEN PRINT A$:PRINT B$:STOP
40 PRINT B$:PRINT A$
```

Fate girare il programma. Al primo INPUT,introducete 'BENE', al secondo 'MALE'. Come potete vedere dal listato sopra, il programma fa sì che il computer stampi per prima la parola più grande, seguita dalla stringa più piccola. La risposta è evidente:

```
MALE
BENE
```

Per il microprocessore il male è più grande del bene! Se voi battete prima 'MALE' e poi 'BENE', il risultato non cambia. Provate ora con altre coppie di parole tipo 'QUESTO' e 'QUELLO', 'TUTTO' e 'NULLA', 'PROGRAMMATORE' e 'PROGRAMMABILE' ed alcune altre. Siete in grado di dire in che modo il computer decide quale è maggiore?

Si tratta di un programma molto semplice per ordinare due parole in ordine alfabetico. La macchina paragona due parole e prende quella che

```
10 DIM W$(21)
20 W$(1) = "ZZZ"
30 PRINT"♥"
40 PRINT:INPUT"PAROLA";X$
50 PRINT"♥":J = 1
60 IF X$ > W$(J) THEN J = J + 1:GOTO 60
70 FOR K = 21 TO J + 1 STEP -1
80 W$(K) = W$(K-1)
90 NEXT K
100 W$(J) = X$
110 N = N + 1
120 X = FRE(0)
130 FOR L = 1 TO N
140 PRINT W$(L)
150 NEXT L
160 IF N < 21 THEN PRINT:GOTO 40
170 PRINT:PRINT"■LISTA COMPLETA";
180 GOTO 180
```

Listato 8.3 - Un programma che mette dei nomi in ordine alfabetico.

in ordine alfabetico viene per seconda come la più grande. Provate il programma con una coppia del tipo 'ANT' e 'ANTARTIC'. Queste parole vengono visualizzate con ANT per primo e ANTARTIC secondo esattamente come facciamo noi per l'ordine alfabetico. Logicamente questo tipo di ordinamento è molto utile per maneggiare liste di nomi di persone, cataloghi e indici di ogni genere.

Il programma nel listato 8.3 contiene una lista di parole in una matrice W$. Quando una nuova parola viene aggiunta, il computer cerca all'interno della lista e lo inserisce nell'esatta posizione. Questo è effettivamente un programma dimostrativo, ma le sue principali routines potrebbero essere modificate ed adattate per renderlo di uso pratico. Esso illustra alcune tecniche di programmazione.

Il programma comincia introducendo un vettore W$, capace di 21 parole ignorando W$(0), ne vedremo il motivo tra un momento). Poi una 'parola conclusiva' ('ZZZ') viene introdotta nella matrice al primo posto (linea 20). Dopo che avete introdotto una parola (X$), il computer esamina una per volta tutte le parole contenute nella lista. La variabile J viene utilizzata per indicare la parola che viene osservata. Alla linea 60 il computer paragona X$ con W$(J), per vedere se in ordine alfabetico X$ viene prima di W$(J). Se sì, J viene incrementato ed il confronto è ripetuto con la parola successiva nella lista. Si procede così finchè X$ risulta non maggiore di W$(J). Ciò può significare sia che X$ è uguale a W$(J) o che viene prima di esso.

Il passo successivo consiste nell'inserire X$ nella lista, in posizione J. Questa operazione viene realizzata traslando tutte le parole rimanenti (dal posto J+1 fino alla fine del vettore) di una posizione. La figura 8.2 mostra quello che succede. Il programma inizia dal fondo della lista, trasferendo di una posizione, (W$(K)), la parola memorizzata nella posizione precedente, (W$(K-1). Poi X$ viene memorizzata in W$(J).

La linea 110 tiene conto del numero di parole nella lista e, nelle linee tra 130 e 150, le parole vengono stampate in ordine sul video, così che voi potete vedere come procede l'ordinamento. La linea 120 comprende un'istruzione BASIC che non è ancora stata menzionata. L'uso più frequente di essa è per cercare e visualizzare quanta memoria vi rimane disponibile per il vostro programma. Se battete:

```
PRINT FRE(0)
```

e premete RETURN il computer visualizza il numero di byte di memoria ancora disponibili per programmare. Lo zero dopo FRE è dummy (fittizio, convenzionale); ci deve essere ma non ha alcun significato o scopo. Imponendo l'istruzione FRE si ordina al VIC di guardare in memoria ed osservare quanto spazio libero c'è. Facendo questo, potrebbe trovare

molto spazio occupato da stringhe che non vengono utilizzate da tempo. Va quindi ad una routine che riorganizza lo spazio di memoria in cui vengono immagazzinate le stringhe, liberandosi di quelle inutilizzate e facendo così spazio in memoria per i programmi. Questo si rende necessario

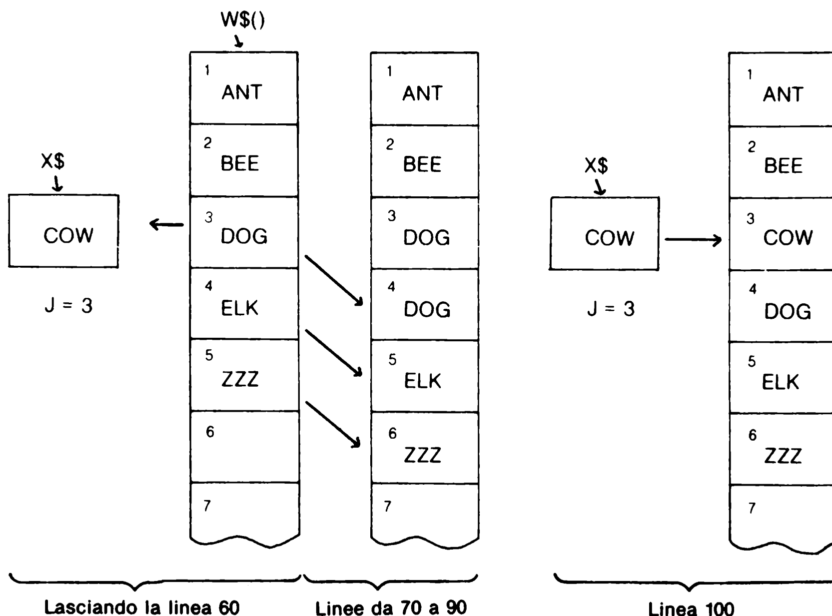


Fig. 8.2 - Inserzione di un nuovo elemento X$ nella sua corretta posizione all'interno di W$

quando si muovono molte stinghe e le si ribattezza più volte, poichè la memoria occupata per immagazzinarle coi nomi vecchi (ossia prima di essere ribattezzate) è sempre occupata. FRE ripulisce tutti questi bit, permettendo di manipolare e memorizzare molte stringhe senza provocare un errore del tipo 'OUT OF MEMORY'. La funzione FRE non è rigorosamente necessaria in questo programma, poichè c'è tutto lo spazio necessario per spostare ed immagazzinare venti parole, ma è stata inserita per ricordarvene l'utilità nei programmi di ordinamento in cui si lavora su un gran numero di elementi.

Ordinando dei numeri

Se il programma precedente viene modificato leggermente, può essere utilizzato per ordinare numeri invece di parole. I cambiamenti necessari sono:

Togliere il simbolo '$' laddove compare
modificare la linea 20 in: W(1)=1E38 (circa il più grande numero che la macchina può gestire)
modificare la linea 40 in PRINT:INPUT'NUMBER';X

Il programma ordina inserendo valori uno alla volta in una lista che è già in ordine. Questa procedura si rende necessaria abbastanza raramente quando si lavora sui numeri. Più spesso si parte da un set di numeri del tutto disordinato e lo si vuole riordinare. Il programma nel listato 8.4 ci mostra la strada più semplice.

```
10 PRINT"□"
20 INPUT"QUANTI NUMERI";N
30 DIM A(N)
40 FOR J = 1 TO N
50 INPUT A(J)
60 NEXT J
70 FOR J = 1 TO N-1
80 FOR K = J + 1 TO N
90 IF A(K) >= A(J) THEN GOTO 130
100 T = A(K)
110 A(K) = A(J)
120 A(J) = T
130 NEXT K
140 NEXT J
150 PRINT"□"
160 FOR J = 1 TO N
170 PRINTA(J)
180 NEXT J
```

Listato 8.4 - Ordinamento di numeri con il metodo degli interscambi.

Il programma inizia chiedendo quanti numeri devono essere ordinati, poi chiede all'utente di introdurli uno alla volta. Questi, ovviamente, possono essere battuti in qualsiasi ordine. Vengono caricati in un vettore A.

La figura 8.3 mostra come lavora questo metodo di ordinamento. Contiene due iterazioni una nell'altra. Al primo giro, quando J=1, il primo numero viene confrontato con tutti gli altri che lo seguono, usando il ciclo interno che richiama tutti i numeri da A(K) ad A(N). Nell'espressione A(K), K vale J+1 così che quando J=1, il primo numero richiamato è A(2). Al secondo passaggio sul ciclo esterno il primo numero chiamato è A(3), e così via per tutti i giri successivi del ciclo. I numeri vengono paragonati con il primo numero alla linea 90. Se A(K) è maggiore o uguale al primo numero, non cambia nulla; il successivo A(K) viene preso per il confronto. Ma se A(K) è minore del primo numero, i due vengono scambiati fra loro.

L'interscambio richiede una variabile T, per la memorizzazione temporanea. La figura 8.4 mostra come agiscono le linee da 100 a 120.

Dopo aver trovato il più piccolo dei numeri ed averlo posto nella prima

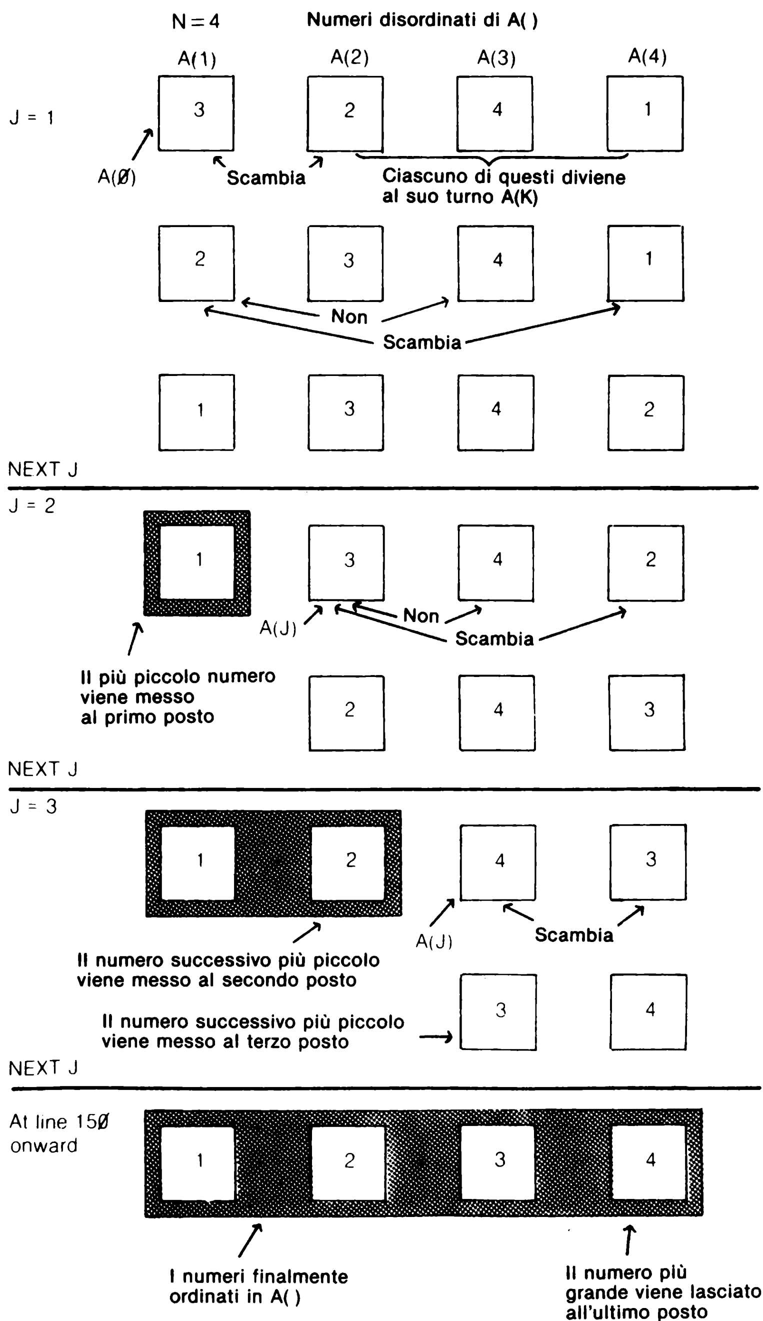


Fig. 8.3 - Funzionamento del metodo degli interscambi; il contenuto di A() viene mostrato di volta in volta dopo ogni scambio. 'Non' significa 'non scambia'.

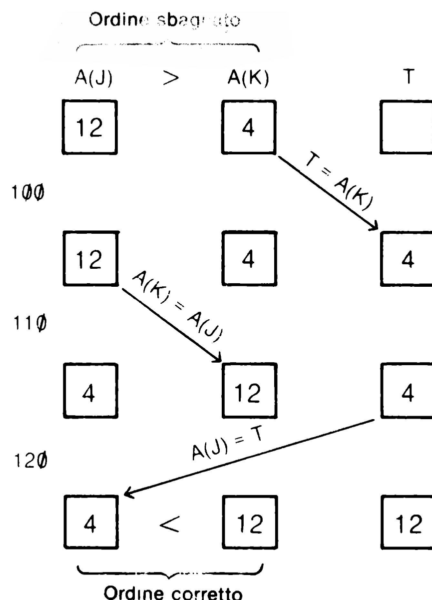


Fig. 8.4 - Uso della variabile temporanea T per scambiare due numeri tra A(J) ed A(K) nei metodi dell'interscambio e della bolla d'aria.

posizione della matrice, la macchina lo ignora. Il ciclo esterno viene ripetuto con J = 2. Ora è il secondo numero ad essere paragonato con tutti quelli che lo seguono (da A(3) in poi) e, come prima, il minore viene posto in seconda posizione. Al terzo giro il più piccolo dei restanti numeri (da A(4) in poi) viene messo al terzo posto. Alla fine il numero più grande è rimasto in n-esima posizione. Non è necessario ripetere l'iterazione N volte, in quanto il numero al posto N è già stato messo a posto durante la (N-1)-esima ripetizione del ciclo esterno.

Fate girare il programma alcune volte introducendo numeri diversi da ordinare. Lavora abbastanza velocemente quando è applicato ad una mezza dozzina di numeri circa, ma già con una ventina impiega un tempo considerevole. Il problema è che il computer deve prendere un numero alla volta e confrontarlo con tutti i successivi. Più sono i numeri nella lista, più sono i numeri 'successivi' da confrontare. Ci vuole un tempo quattro volte superiore per ordinare un vettore di lunghezza doppia.

Potete osservare il funzionamento del programma più da vicino aggiungendo queste linee:

```
125 FOR L=1 TO N:PRINT A(L);:NEXT L
126 PRINT' '
135 PRINT'*';
```

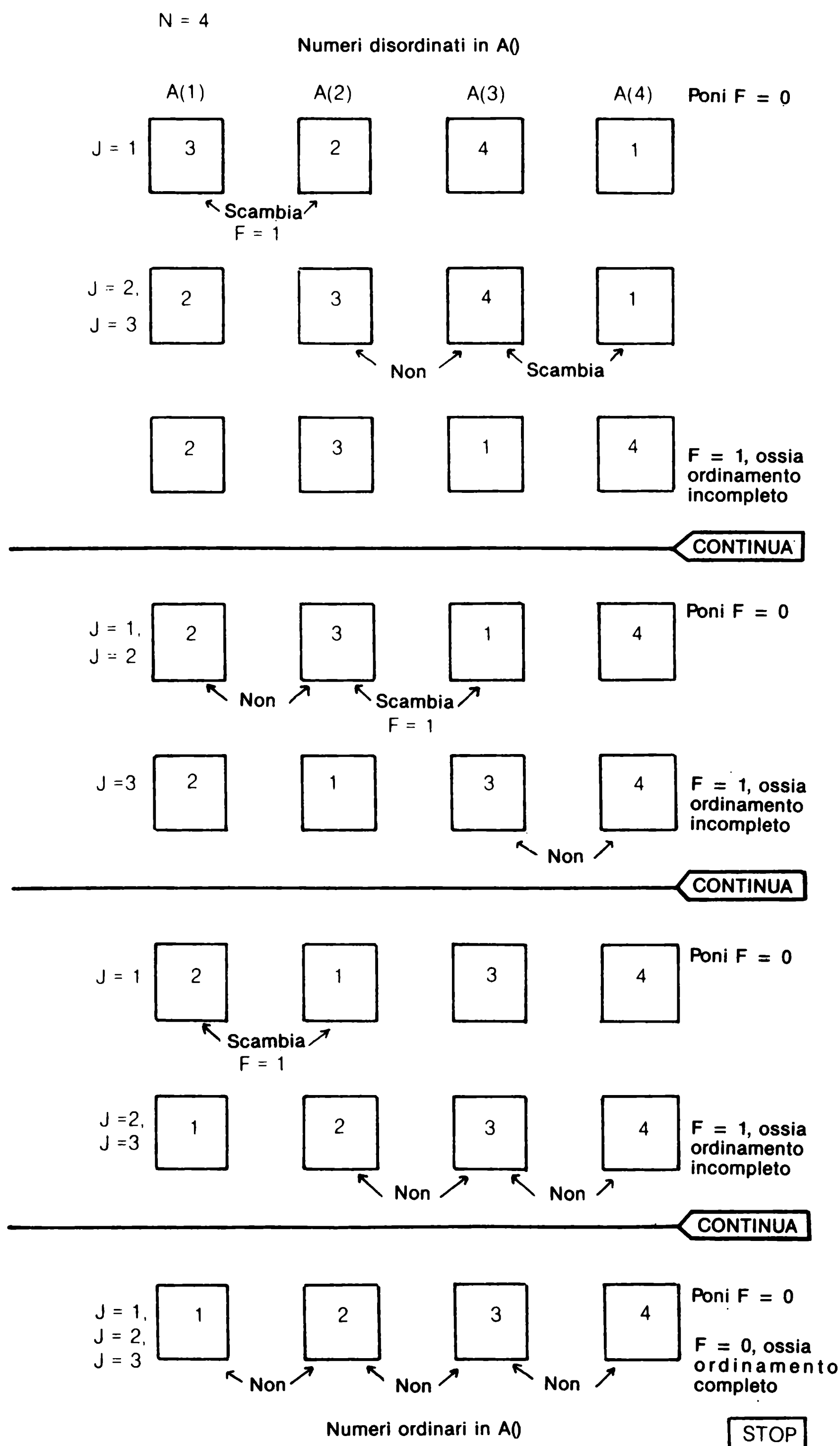


Fig. 8.5 - Funzionamento del metodo della bolla d'aria; il contenuto di A() viene mostrato dopo ogni scambio. 'Non' significa 'non scambia'.

```
10 PRINT"♥"
20 INPUT"QUANTI NUMERI";N
30 DIM A(N)
40 FOR J = 1 TO N
50 INPUT A(J)
60 NEXT J
70 J = 1
80 F = 0
90 IF A(J) <= A(J+1) THEN GOTO 130
100 T = A(J)
110 A(J) = A(J+1)
120 A(J+1) = T
125 F = 1
130 J = J + 1:IF J< N THEN GOTO 90
140 IF F = 1 THEN GOTO 70
150 PRINT"♥"
160 FOR J = 1 TO N
170 PRINT A(J)
180 NEXT J
```

Listato 8.5 - Ordinamento di numeri col metodo della bolla d'aria.

Notate il punto e virgola nelle linee 125 e 135. Cancellate la linea 150. Ora il video vi mostra il risultato di ogni interscambio. Stampa anche un '*' ogni volta che arriva alla fine del ciclo esterno. Se vedete due o più asterischi insieme, vuol dire che non vi è stato nessuno scambio durante quel ciclo esterno. La macchina ha perso tempo ad esaminare numeri che erano già nel giusto ordine. Questo semplice programma può essere anche utilizzato per ordinare parole. Basta correggere le variabili A() in A$() e T() in T$().

Metodo della bolla d'aria (bubble sort)

Questo programma è leggermente più complicato di quello dell'interscambio, ma può essere molto più veloce. Il suo funzionamento è mostrato in fig. 8.5. Il computer lavora facendo una scansione del vettore prendendo i numeri a coppie. Quando trova una coppia nell'ordine sbagliato, ne inverte l'ordine . Poi torna indietro e rifà la stessa operazione. Ogni volta che comincia a scandire il vettore verso il basso, fissa a 0 un flag, F. Se fa uno scambio, manda F ad 1. Se passa in rassegna tutto il vettore e non si rende necessario nessun cambiamento, i numeri devono essere nel giusto ordine e il procedimento è terminato. Se in partenza i numeri sono abbastanza in ordine, vengono rapidamente sistemati. Una volta che il lavoro è fatto il flag dice al computer di fermarsi senza sprecare tempo per operazioni inutili. Venti numeri di un solo vettore, in un ordine più o meno corretto, vengono ordinati in maniera estremamente rapida. D'altro canto, se sono quasi in ordine inverso il bubble sort impiega più tempo del metodo degli interscambi descritto nel paragrafo precedente. Nel me-

todo della bolla d'aria i numeri più piccoli risalgono la cima del vettore gradualmente, come le bolle d'aria risalgono in un liquido. Questa è la caratteristica che dà il nome al metodo.

Come nel programma degli interscambi, due valori vengono invertiti utilizzando una variabile temporanea (fig. 8.4). Le tre linee che realizzano questo (dalla linea 100 alla 120) potrebbero essere conglobate in una linea multi-comando. Questo farebbe girare il programma un po' più velocemente. Anche il bubble sort, così come l'interscambio, può essere adattato ad ordinare parole.

Riassunto

In questo capitolo avete visto come:

— usare le funzioni LEFT$, RIGHT$ e MID$ per maneggiare parti di stringhe
— usare le funzioni STR$ e VAL per convertire numeri in stringhe e stringhe in numeri
— usare le funzioni LEN e FRE(0)
— mettere parole in ordine alfabetico, usando gli operatori di correlazione < e >
— ordinare parole o numeri usando il metodo degli interscambi
— ordinare parole o numeri usando il metodo bubble sort.

Capitolo 9
Ulteriori informazioni sui suoni

Abbiamo già visto in dettaglio nel capitolo cinque i generatori di suono del VIC e come utilizzarli. Questo capitolo vi mostra un modo veloce per per creare effetti acustici speciali da aggiungere ai vostri programmi.

Una delle cose più difficili da realizzare negli effetti sonori è l'impossibilità di prevedere come suoneranno, una volta che verrà fatto girare il programma. Se volete, per esempio, ottenere il suono della sirena della polizia, che numero dovete introdurre ed in quale generatore? Per quanto tempo deve protrarsi? Si può ottenere una risposta soddisfacente a queste domande solo provando alcuni gruppi di valori ed ascoltandone i risulta-

```
10 R=36874:PRINT"♥":C$="DDLLLMMMHHHNNNVV"
20 FOR J = 1 TO 5
30 PRINTTAB(2)J;C$
35 PRINTTAB(2)J;"**";RIGHT$(C$,14):NEXT J
40 PRINTTAB(5)C$:PRINT"◘"
60 INPUTN$(1),N$(2),N$(3),N$(4),N$(5)
65 INPUTN$(6),N$(7),N$(8),N$(9),N$(10)
70 FOR J = 1 TO 10
80 A(J,0) = VAL(MID$(N$(J),3,2))
90 A(J,1) = VAL(MID$(N$(J),5,3))
100 A(J,2) = VAL(MID$(N$(J),8,3))
110 A(J,3) = VAL(MID$(N$(J),11,3))
120 A(J,4) = VAL(MID$(N$(J),14,3))
130 A(J,5) = VAL(RIGHT$(N$(J),2))
140 NEXT J
150 FOR J = 1 TO 9 STEP 2
160 D = A(J,0):IF D = 0 THEN GOTO 220
170 FOR K = 1 TO D
180 FOR L = 1 TO 5
190 POKER+L-1,A(J,L)+(A(J+1,L)-A(J,L))*K/D
200 NEXTL:NEXTK
220 NEXTJ
230 FOR L=0 TO 4:POKE R + L,0:NEXT L
240 PRINT"◘"
250 GOTO 60
```

Listato 9.1 - FXG, un programma utility sugli effetti sonori che vi aiuterà a preparare gli effetti adatti ai vostri programmi.

ti. Poi diviene semplice individuare il o i valori che devono essere corretti. Non è necessario toccare gli altri. L'unico problema è che bisogna ricordarsi di reinserirli sempre tutti.

Il programma FXG (abbreviazione di 'Generatore di Effetti') vi permette (guardate il Listato 9.1) di provare più volte un segnale sonoro, correggendo uno o più valori, ma lasciando gli altri inalterati. Il video mostra un elenco di tutti i parametri che state utilizzando, così che potete chiaramente vedere quello che state facendo. Quando avrete ottenuto il suono desiderato vi basterà copiare i valori visualizzati ed introdurli nel vostro programma.

Come i programmi per generare caratteri del capitolo sette, FXG è un programma di per sé piuttosto complicato. Ciò è inevitabile, in quanto affida al microprocessore parte del lavoro che avreste dovuto fare voi manualmente. Se volete semplicemente inventare dei vostri effetti acustici personali, ma avete poco interesse a come funziona il programma, passate direttamente al paragrafo 'Usare l'FXG', dove vengono illustrate le istruzioni da dare per utilizzarlo. Per coloro i quali, invece, sono più seriamente interessati alla programmazione, ecco una breve descrizione di come funziona il programma.

				Durata		Note basse (bassi)			Note medie (tenori)			Note acute (soprano)			Rumore			Volume	
Fase 1	Valori iniziali →	1		D	D	L	L	L	M	M	M	H	H	H	N	N	N	V	V
	Valori finali →	1		*	*	L	L	L	M	M	M	H	H	H	N	N	N	V	V
Fase 2		2		D	D	L	L	L	M	M	M	H	H	H	N	N	N	V	V
		2		*	*	L	L	L	M	M	M	H	H	H	N	N	N	V	V
Fase 3		3		D	D	L	L	L	M	M	M	H	H	H	N	N	N	V	V
		3		*	*	L	L	L	M	M	M	H	H	H	N	N	N	V	V
Fase 4		4		D	D	L	L	L	M	M	M	H	H	H	N	N	N	V	V
		4		*	*	L	L	L	M	M	M	H	H	H	N	N	N	V	V
Fase 5		5		D	D	L	L	L	M	M	M	H	H	H	N	N	N	V	V
		5		*	*	L	L	L	M	M	M	H	H	H	N	N	N	V	V

Battete «RETURN» solo quando il cursore si trova su questa colonna

Non immettete nulla al posto di *

Estremità di destra dello schermo

Fig. 9.1 - Il display del programma FXG. Le linee di lettere vi mostrano dove introdurre i valori relativi agli effetti acustici.

La linea 10 fissa R come indirizzo base dei generatori di suoni del VIC. C$ viene utilizzato nelle linee da 20 a 40, che visualizzano il display (fig. 9.1). Dopo che il display è stato visualizzato, la linea 50 manda il cursore nella posizione 'home'. Ora l'INPUT della linea 60 fa apparire un punto di domanda in alto sullo schermo, appena sopra il display. Il computer attende a questo punto i dati in ingresso.

Una delle caratteristiche più inconsuete del VIC è che assume i valori in ingresso direttamente dal video. Gran parte degli altri microcomputer usano invece una parte di memoria chiamata 'input buffer'. Tutto quello che viene introdotto nella macchina viene prima memorizzato in questo buffer. Esso può essere o meno visualizzato sullo schermo. Quando l'utente ha battuto la risposta al comando di INPUT ed ha premuto RETURN, molti degli altri microprocessori vanno all'input buffer a cercare ciò che è stato introdotto. Il VIC invece va allo schermo RAM e legge quello che viene visualizzato sullo schermo. Nel programma FXG viene fatto pieno uso di questa caratteristica del VIC. Il cursore viene sempre rimandato nella posizione di home immediatamente prima di ogni comando di INPUT. Voi potete così spostarlo tra i caratteri visualizzati sullo schermo usando il tasto di comando destra-sinistra. Battete RETURN alla fine di ogni linea sul video. I caratteri sui quali è già passato il cursore vengono letti come INPUT. Il vantaggio di questa tecnica è che quando volete correggere solo pochi dettagli degli effetti acustici, tutto ciò che dovete fare è cambiare alcuni parametri sullo schermo, battendo i nuovi valori su di essi. Tutti i valori che non vengono corretti restano inalterati sullo schermo e vengono letti come INPUT semplicemente passandoci sopra col cursore. Le linee in cui non sono state fatte modifiche vengono scandite col cursore ed alla fine si batte RETURN. Quando sono coinvolti tanti parametri e se ne devono modificare solo alcuni, questa tecnica è molto pratica perchè fa risparmiare tanto tempo.

Le dieci righe del display vengono lette separatamente nella matrice N$().

J ↓	Ø	1	2	3	4	5		
Ø	Ø	Ø	Ø	Ø	Ø	Ø	Non utilizzati	
1							Valori iniziali	Fase 1
2							Valori finali	
3	Durata della fase	Note basse	Note medie	Note acute	Rumore	Volume	Fase 2	
4								
5							Fase 3	
6								
7							Fase 4	
8								
9							Fase 5	
1Ø								

Fig. 9.3 - La matrice A(), utilizzata per memorizzare i valori degli effetti sonori.

Nelle linee da 80 a 130, porzioni di ogni elemento di N$ vengono convertite in numeri (usando VAL) e messe in un'altra matrice (A). I contenuti di A sono illustrati in fig. 9.2.

Il programma passa poi in rassegna A(), leggendo le righe con etichetta dispari per vedere quali valori iniziali bisogna dare ai generatori di suono all'inizio di ciascuna fase del suono. Le linee di A() con etichetta pari contengono i valori finali. La linea 160 guarda se la durata del suono è zero o, in altri termini, se c'è qualche fase del suono che deve essere saltata. Le linee da 180 a 200 assegnano di volta in volta ai generatori ed ai rispettivi controlli di volume i valori presi da A(). I valori da introdurre ad ogni livello dipendono dal valore iniziale, da quello finale, dal valore che K assumerà nella linea 170 (assegnato da D) e dal valore attuale di K. Dalla fig 9.3, risulta evidente che la formula necessaria è:

$$\text{VALORE} = \text{INIZIALE} + \frac{[(\text{FINALE} - \text{INIZIALE})^*\text{K}]}{\text{D}}$$

ove K è l'attuale valore che K assume durante ogni iterazione. Questo calcolo viene fatto per ogni gruppo di valori proveniente da A(), nella linea 190.

Quando l'effetto acustico è terminato, tutti i generatori ed i controlli di volume vengono portati a zero nella linea 230. Il cursore viene poi rimesso in posizione 'home' ed il programma ritorna alla linea 60 per ricevere nuovi INPUT.

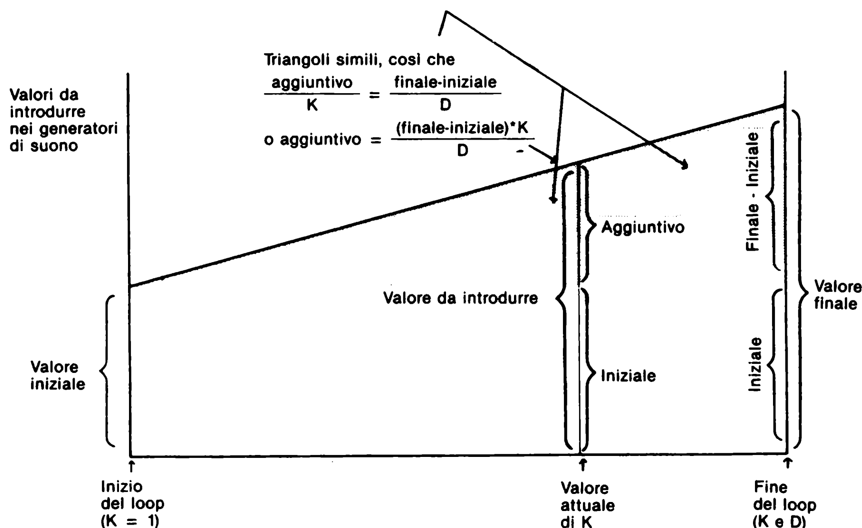


Fig. 9.3 - Come il computer calcola i valori corrispondenti alle note che crescono o decrescono.

Usare l'FXG

L'idea di questo programma è far generare al computer un effetto sonoro che può avere fino a cinque fasi. Per esempio, prendete l'effetto sonoro usato nel programma Raid Aereo (Listato 5.4). La fase 1 simula il sibilo di una bomba mentre cade, che cresce gradualmente in volume. La fase 2 utilizza il generatore di rumore solo al massimo volume per riprodurre l'esplosione. Nella fase 3 il rumore scompare per essere sostituito nella fase 4 dal basso ronzio dei motori degli aerei. Per ogni fase dovete decidere quanto tempo deve durare, quali generatori utilizzare, quali parametri assegnare all'inizio (valori iniziali) e alla fine della fase (valori finali). Dovete inoltre stabilire l'andamento del volume durante ciascuna di esse. Queste sono le operazioni da eseguire per usare il programma:

(1) Fate girare il programma (RUN). Il display appare sul video come in fig. 9.1, con un punto di domanda sopra di esso sulla sinistra.
(2) Premete il tasto di spostamento del cursore alto-basso una sola volta per portare il cursore nella prima linea del display.
(3) Spostate il cursore sulla cifra '1' all'inizio della prima linea, usando il tasto di comando destra-sinistra del cursore e muovetelo verso destra finchè siete sulla prima delle due 'D'. Qui dovete scrivere la durata della prima fase. Deve essere un numero composto da due cifre. Per cominciare provate con '10'. Questo fa durare la fase esattamente 1 secondo. Se volete avere una fase più corta, scrivete un numero minore di '10', ma battete sempre uno zero iniziale (per es. '07'). Se fate un errore, spostate indietro il cursore, premendo SHIFT ed il tasto per spostare a destra-sinistra il cursore, finchè siete sulla cifra sbagliata. Ribattete poi la cifra correttamente.
(4) Procedete sulla linea battendo cifre al posto di:

LLL - per il generatore dei toni bassi (basso)
MMM - per il generatore dei toni medi (tenore)
HHH - per il generatore dei toni alti (soprano)
NNN - per il generatore di rumore

Non è necessario che usiate tutti questi, basta infatti che ne usiate uno (ad esempio il generatore dei toni medi) e lasciate inalterate le caselle di LLL, HHH e NNN. I valori introdotti devono essere nel dovuto intervallo, tra 128 e 254. Alla fine portate il cursore su VV e battete il volume richiesto (da 01 a 15).
(5) Il cursore si trova ora su una casella vuota all'estrema destra della prima linea (fig. 9.1). Battete RETURN che fa passare il cursore alla seconda linea. Ora appare sopra l''1'. Due punti interrogativi ('??') appaiono sulla sinistra.

(6) Muovete il cursore oltre i '**'all'inizio della linea. Battete ora i valori finali per quei generatori per i quali avete dato i valori iniziali nella prima linea. Come prima non vi è alcuna necessità di assegnare parametri ai generatori che non intendete usare. Battete RETURN quando il cursore raggiunge lo spazio bianco alla fine della linea.
(7) Se il suono che volete realizzare è composto da più fasi, continuate come nei punti da (3) a (6) appena visti per le linee che cominciano con '2', '3', '4'e '5' secondo il numero di fasi che volete. La linea di lettere più in basso (senza numeri) serve solo per aiutarvi nell'orientamento. Ricordate che è essenziale premere RETURN ogni volta che il cursore arriva allo spazio vuoto alla fine di ogni linea.
(8) Una volta inseriti i parametri dettagliati di ogni fase, saltate alla fine del display premendo ripetutamente RETURN. Non è necessario muovere il cursore lungo le linee che non vengono utilizzate. Dopo aver premuto RETURN per la seconda delle linee '5', vi è una piccola pausa e poi l'effetto sonoro viene generato.
(9) Al termine del suono, riappare il cursore all'angolo in alto a sinistra del video. Per ripetere l'effetto acustico basta premere RETURN.
(10) Per correggere l'effetto, muovete il cursore tra le linee del display, ribattendo tutti i caratteri che desiderate modificare e battendo le nuove cifre che volete eventualmente aggiungere. Battete RETURN alla fine di ogni linea, esattamente come sopra nei punti da (3) a (7). Le cifre che non sono state modificate vengono mantenute nel nuovo effetto sonoro.
(11) Quando siete soddisfatti del suono ottenuto, copiate i paremetri dal video.

Il limite principale del programma FXG viene evidenziato dalle note in scala ascendente o discendente alle frequenze più alte. Questo è dovuto in parte al modo con cui i generatori calcolano la frequenza. All'estremità superiore della scala un incremento o decremento di 2 o 3 nel valore introdotto nei generatori, produce un intervallo di un semitono o, addirittura, di un tono intero. Al posto di un suono che varia con continuità (come otteniamo nel caso delle note basse) abbiamo una successione di note distinte di altezza crescente o decrescente. Uno dei motivi di questo comportamento sta nel fatto che i calcoli richiesti durante il funzionamento del programma impongono che tutti i cambiamenti in altezza vengano realizzati con pochi grossi passi, piuttosto che con tanti piccoli. Se volete controllare due o tre generatori contemporaneamente, questo comportamento è difficile da evitare, ma, se il vostro effetto acustico può essere realizzato usando un generatore per volta, è possibile migliorare il risultato rendendo i passi più piccoli ed aumentandone il numero. Questo viene trattato nel seguente esempio.

Presentiamo ora un esempio per convertire i valori ottenuti da FXG in

linee da inserire in qualche vostro programma. L'effetto che si vuole ottenere è il rumore del motore di una motocicletta che frena ad uno stop col rumore di vetri che si rompono in un incidente. La cosa non è molto piacevole da sperimentare di persona, ma è innocua come effetto acustico! Viene divisa in quattro fasi:
(1) Una nota bassa di altezza crescente prodotta dal generatore basso. Il display visualizzato sullo schermo appare nel Listato 9.2

```
1  30150MMMHHHNNN15
1  **180MMMHHHNNN15
```

Listato 9.2 - Il display di FXG sullo schermo per la prima fase dell'effetto, la motocicletta.

La fase dura circa tre secondi ed è a pieno volume.
(2) Una nota alta di altezza crescente combinata con un rumore anch'esso alto, per simulare lo stridere delle gomme ed i vetri che si rompono. Questa fase è corta e dura meno di un secondo. Il display che appare viene mostrato nel Listato 9.3.

```
2  08LLLMMM25025415
2  **LLLMMM25225415
```

Listato 9.3 - . Il display di FXG sullo schermo per la seconda fase dell'effetto, la brusca frenata.

Come prima, il volume è al massimo.
(3) La simulazione del rumore di una collisione è mostrata nel Listato 9.4.

```
3  06LLLMMMHHH20015
3  **LLLMMMHHH22015
```

Listato 9.4 - Il display di FXG sullo schermo per la terza fase dell'effetto, l'impatto.

Notate l'ampiezza crescente del suono. Vi è molta libertà di azione per modificare i parametri ed ottenere esattamente l'effetto che vi eravate prefissi.
(4) Il rumore dell'incidente che si smorza è ottenuto coi parametri del Listato 9.5.

```
4  15LLLMMMHHH22012
4  **LLLMMMHHH22000
```

Listato 9.5 - Il display di FXG sullo schermo per la quarta fase dell'effetto, lo smorzamento dell'eco dell'incidente.

Il volume finale è zero e impiega esattamente 1.5 secondi per raggiungerlo.
Provate a far girare il programma FXG coi parametri per ottenere questo effetto. Cambiate alcuni valori per migliorarlo; potreste farvi delle idee

differenti sui tempi di durata o sulle ampiezze.

Quando viene utilizzato in un programma completo, questo effetto viene scritto in quattro loop separati, uno per ogni fase. Per rendere il programma più leggibile conviene definire con 4 variabili gli indirizzi dei generatori di suono ed il controllo di volume (vedere il Listato 9.6).

```
10 S1 = 36874:S3 = 36876:N = 36877:V = 3
6878
```

Listato 9.6 - Preparazione delle variabili per l'effetto dell'incidente.

Il generatore dei medi non viene utilizzato e quindi il suo indirizzo non viene definito.

La fase 1 è un loop che dura circa tre secondi. La sua struttura principale viene illustrata nel Listato 9.7.

```
20 POKE V,15
30 FOR J = 1 TO X
40 POKE S1,150 + 30*J/X
50 NEXT J
```

Listato 9.7 - Il programma provvisorio per provare la fase 1.

Il calcolo del valore nella linea 40 utilizza l'equazione mostrata precedentemente in questo capitolo. Il valore iniziale è 150, quello finale 180; la loro differenza dà il valore '30', che viene poi moltiplicato per J e diviso per X (il numero di passi). Prima di provare questo effetto, introducete anche la linea mostrata nel Listato 9.8 per mettere alla fine tutto a tacere.

```
200 POKE S1,0:POKE S3,0:POKE N,0:POKE V,0
```

Listato 9.8 - L'interruzione dei generatori alla fine del programma.

Assegnate un valore ad X scrivendo un'ulteriore linea che verrà poi tolta (vedere il Listato 9.9).

```
25 X = 200
```

Listato 9.9 - Un valore di prova per il loop.

Ora fate girare il programma. Sentite il rumore della motocicletta. Questo sarà lo stesso che avete sentito nel programma FXG, ma potete pensare che la fase non duri abbastanza. Finchè vi è un solo indirizzo da inserire ad ogni giro del loop, questo loop gira molto più velocemente di quello in FXG. Per far sì che il nostro loop giri più lentamente, modificate la

linea 25 come mostrato nel Listato 9.10. Provate ora di nuovo e probabilmente converrete con noi nel dire che il suono è quello giusto. Come indi-

```
25 X = 300
```

Listato 9.10 - Un altro valore di prova.

cazione grossolana, assumete che, quando vi è un solo indirizzo da inserire, il conteggio di 100 corrisponde circa ad un secondo. Un conto fino a 300 impiega quindi circa tre secondi. Ora che abbiamo stabilito che 300 è il valore adatto per X, possiamo inserirlo nel programma e cancellare la linea 25. Il loop così migliorato è mostrato nel Listato 9.11. Notate co-

```
30 FOR J = 1 TO 300
40 POKE S1,150 + J/10
50 NEXT J
```

Listato 9.11 - La versione finale del loop che simula la prima fase.

me '30*J/300' viene semplificato in 'J/10'. Passiamo ora alla fase 2 del programma, la frenata con stridio ed i vetri che si rompono. Questo ha in gran parte la stessa forma del loop nel Listato 9.11, ma usa il generatore di alti invece di quello dei bassi. Prima che il loop cominci, il generatore di rumore viene indirizzato per dare lo stridio. Un punto importante, ricordatevi di fermare il generatore dei bassi!

```
60 POKE S1,0:POKE N,254
70 FOR J = 1 TO X
80 POKE S3,250 + 2*J/X
90 NEXT J
```

Listato 9.12 - Il programma provvisorio per provare la fase 2.

Come prima, viene utilizzata la formula precedentemente mostrata all'inizio del capitolo per calcolare il valore da introdurre. Inseriamo una linea aggiuntiva per poter provare diversi valori di X. Siccome questa fase

```
65 X = 80
```

Listato 9.13 - Un valore di prova per il loop della fase 2.

deve esaurirsi in meno di un secondo, la linea potrebbe essere quella nel Listato 9.14. Ancora '2*J/80' viene semplificato in 'J/40'.

```
70 FOR J = 1 TO 80
80 POKE S3,250 + J/40
90 NEXT J
```

Listato 9.14 - La versione finale della seconda fase.

La fase tre richiede che il generatore di rumore cresca in ampiezza mentre il volume decresce lentamente, da 15 a 12. Prima dell'inizio del loop, bisogna fermare il generatore degli alti (vedere il Listato 9.15).

```
100 POKE S3,0
110 FOR J = 1 TO X
120 POKE N,220 + 20*J/X
130 POKE V,15 - 3*J/X
140 NEXT J
```

Listato 9.15 - Un programma provvisorio per provare la fase 3.

La linea 105 viene utilizzata per trovare un buon valore di X. 60 è qui un po' troppo elevato in quanto ora vi sono due parametri da inserire nel loop. Un valore di 30 o 40 sembra ragionevole. Provatelo introducendo:

```
105 X = 30
```

Listato 9.16 - Un valore di prova per il loop della fase 3.

Se vi sembra corretto, modificate le linee da 110 a 140 come mostrato nel Listato 9.17. Ancora una volta i valori da introdurre si ottengono sostituendo i valori prescelti delle X nelle rispettive espressioni e cancellando dove necessario.

```
110 FOR J = 1 TO 30
120 POKE N,220 + 2*J/3
130 POKE V,15 - J/10
140 NEXT J
```

Listato 9.17 - La versione finale della terza fase.

La fase finale mantiene il generatore di rumore ad un'ampiezza costante, ma il volume diminuisce fino a zero. La fase precedente lascia il volume già inizializzato a 12 all'inizio di questa fase ed il generatore di rumo-

```
150 FOR J = 1 TO X
160 POKE V,12 - 12*J/X
170 NEXT J
```

Listato 9.18 - Un programma provvisorio per provare la fase 4.

re inizializzato a 220, così che la programmazione di questa fase è più semplice (vedere il Listato 9.19).

```
145 X = 150
```

Listato 9.19 - Un valore di prova per il loop della fase 4.

Questo dà un risultato molto valido e le linee vengono dunque corrette come mostrato nel Listato 9.20.

```
150 FOR J = 1 TO 150
160 POKE V,12 - 2*J/25
170 NEXT J
```

Listato 9.20 - La versione finale della quarta fase.

L'effetto originariamente creato da FXG è stato così convertito in linee, che possono essere inserite in qualsiasi programma BASIC nel punto dove si vuole ottenere il suono. Se questa necessità si ripete più volte, è meglio realizzare l'effetto in una subroutine. Essa deve poi terminare con RETURN.

Questo esempio vi ha mostrato come un effetto, delineato con FXG, può essere poi incorporato nei vostri programmi. FXG è un utile programma, ma è stato reso il più corto possibile così che non richiede troppo tempo per inserirlo. Quando il suo utilizzo vi sarà più familiare, potrete aggiungergli dei miglioramenti personali. Un'utile aggiunta potrebbe essere permettere all'utente di far ripetere indefinitamente il suono. Questo rende più agevole la creazione di effetti ripetitivi, quali rumori di macchinari, locomotive, cinguettii, sirene e così via. Potreste anche aumentare il numero di fasi che è possibile includere. L'uso di FXG vi aiuterà ad entrare nell'affascinante mondo degli effetti sonori del VIC. Avendo scelto i vostri effetti utilizzando questo programma, troverete poi facile fare sottili modifiche per perfezionarli.

Riassunto

In questo capitolo avete trovato:

— come creare nuovi effetti sonori, utilizzando il programma FXG
— come incorporare i vostri effetti in un programma in BASIC.

Capitolo 10
Le funzioni

Diamo per scontato che il vostro VIC funzioni effettivamente, nel senso che si comporta nel modo previsto e rivolgiamo ora lo sguardo ad alcune delle sue funzioni. La parola 'funzione' viene utilizzata troppo spesso nella letteratura informatica. Come potete vedere, dall'inizio del capitolo abbiamo già utilizzato 'funzione' ben cinque volte! Si tratta di un termine con un vasto significato, che ricopre quasi tutto ciò che significa 'fare qualcosa', così che è stato facile per gli informatici sfruttarlo in un gran numero di modi quasi privi di significato e non collegati tra loro.

I tasti funzionali

Si tratta di quattro tasti marroncini sulla destra della tastiera. La loro funzione (scusate, il loro compito) è di permettervi di interagire col programma mentre sta girando. Questi tasti si usano battendone uno mentre il computer sta svolgendo l'istruzione GET. Il tasto RETURN non deve essere premuto.

```
10 N = 1
20 PRINT"□■":POKE 36879,59
30 GET A$:IF A$ <> CHR$(133) THEN GOTO 30
40 FOR J = 1 TO 110
50 PRINTTAB(J)"  ▙▞▜";
60 GET A$
70 IF A$=CHR$(136) AND POS(0)=12 THENGOTO 110
80 PRINT"□"
90 NEXT J
100 N = N + 1:GOTO 40
110 PRINT"□":POKE 36879,43:M = M +1
120 FOR K = T TO 100:NEXT K
130 POKE 36879,59:PRINT"■VOLI";N;"COLPI";M
140 GET A$:IF A$ <> CHR$(134) THEN GOTO 140
150 GOTO 20
```

Listato 10.1 - Target, un gioco che mostra come utilizzare i tasti funzionali.

I tasti funzionali sono particolarmente utili per i giochi. Possono essere utilizzati per fare sparare i razzi, per controllare i movimenti di un missile o di un avventuriero in un labirinto. Hanno anche applicazione in programmi di altro genere. Per esempio, nel word processing (elaborzione di testi) possono essere usati per dare certi comandi sulle forme dei testi.

Il Listato 10.1 è un programma che mostra come questi tasti funzionali possano essere utilizzati.

Mentre il programma gira, un aeroplano vola per lo schermo più volte. Può essere abbattuto solo quando si trova esattamente al centro del video.

La maggior parte delle caratteristiche del programma sono già state descritte nei primi capitoli; ci concentreremo quindi solo su come bisogna usare questi tasti funzionali. Ad ogni modo vi è in questo programma una nuova funzione (ancora quella parola!) che non è ancora stata illustrata.

Si tratta di POS, che appare nella linea 70. Ha come argomento '0', ma questo è dummy e non importa quale valore assuma, in quanto non ha alcun significato. POS assume il valore della colonna del video nella quale apparirà il prossimo carattere da stampare. In questo programma lo utilizziamo per controllare di quanto si è mosso l'aereo nel suo volo sullo schermo. Quando il naso del velivolo arriva alla colonna 11, la successiva posizione che sarà occupata è la colonna 12 e POS(0) dà il valore 12. Quando si trova in questa posizione l'aereo può essere abbattuto.

Consideriamo ora i tasti funzionali. La linea 30 contiene l'istruzione GET A$. Quando il computer arriva a questa linea, fa una scansione di tutta la tastiera per vedere se in quell'istante viene premuto qualche tasto.

In caso contrario, A$ viene posto ' ' (una stringa vuota).Se invece ne state premendo uno, A$ diventa una stringa composta da un solo carattere, che contiene ciò che stavate premendo. La linea 30 paragona A$ con il codice ASCII dei caratteri, facendo uso della funzione CHR$. Nonostante il fatto che i tasti funzionali non fanno comparire nulla sullo schermo, ciascuno di essi è associato ad una coppia di codici ASCII, a seconda che venga usato o meno il tasto di shift, come viene mostrato nella Tavola 11 dell'appendice A. Siccome non si tratta di codici ASCII standard è meglio far loro riferimento come codici CHR$. Il codice per la funzione F1, senza lo shift, è 133. Il computer ripete la linea 30 continuamente, finchè non viene premuto il tasto F1 (senza lo shift). Il risultato è che, mentre gira il programma, lo schermo rimane di un chiaro blu turchese (ciano), finchè non viene premuto il tasto F1. In quell'istante l'aeroplano inizia il volo. Viaggia da sinistra verso destra e, una volta uscito dallo schermo a destra, riappare sulla sinistra una riga più in basso. Alla fine di questo 'volo', riprende di nuovo dalla riga più alta. Il numero di voli viene memorizzato nella variabile N.

Alla linea 70 la tastiera viene controllata nuovamente. Questa volta il computer sta vedendo se viene premuto o meno il tasto F7 (il cui codice CHR$ è 136). Se il giocatore preme F7 esattamente nell'istante in cui l'aereo arriva alla colonna 11, questo viene abbattuto.

Il programma salta allora alla linea 110, dove fa lampeggiare lo schermo di rosso. Poi viene mostrato il punteggio. La linea 140 mantiene il display finchè non premete il tasto F3, che fa ripartire il programma.

Per giocare, quindi, la sequenza di istruzioni è:

(1) Fate girare il programma.
(2) Quando siete pronti ad iniziare il gioco, battete F1.
(3) Per sparare all'aeroplano, premete F7. Cercate di farlo esattamente quando l'aereo raggiunge il centro dello schermo.
(4) Per cancellare il punteggio dal video, pronti per ricominciare, battete F3.

Ecco un'altra possibilità di utilizzare gli effetti sonori; aggiungete il ronzio dei motori dell'aereo e il rumore dell'esplosione quando lo colpite.

Funzioni del BASIC

Molti dei termini che si usano nei programmi in BASIC vengono chiamati funzioni. Nuovamente notate l'accezione 'attiva' del termine. Una funzione prende una variabile e con questa 'fa qualcosa'. Generalmente la variabile viene posta fra parentesi e viene detta argomento della funzione. Il capitolo otto descrive l'operato delle funzioni di stringa, che 'fanno qualcosa' alle stringhe. Vi sono inoltre molte funzioni numeriche, come INT o RND. INT, per esempio, prende il numero o la variabile che si trova al suo argomento e toglie tutte le cifre dopo la virgola.

Altra funzione del BASIC è SQR, che calcola la radice quadrata di un numero. Provate a fare:

```
10 INPUT X
20 PRINT SQR(X)
```

Questo immediatamente visualizza le nove cifre più significative della radice quadrata di X. Se introducete un numero negativo, la macchina vi indica errore ('?ILLEGAL QUANTITY ERROR IN 20'), in quanto la radice di un numero negativo è una quantità immaginaria, cosa che il VIC non conosce.

Numerose funzioni lavorano con gli angoli. Quelle disponibili sul VIC sono:

SIN(X) che dà il seno dell'angolo X.
COS(X) che dà il coseno dell'angolo X.
TAN(X) che dà la tangente dell'angolo X.
ATN(X) che dà l'arcotangente dell'angolo X.

TAN ed ATN sono una l'opposta dell'altra. In tutte queste funzioni, gli angoli vengono misurati in radianti e non in gradi, ai quali la maggior parte di voi sarà abituata. Se volete usare queste funzioni in un programma e volete che l'utente possa inserire i valori in gradi, o riceva risposte in gradi, dovete ricordarvi di convertire da gradi a radianti o da radianti in gradi al momento più opportuno nel programma. Un semplice modo per farlo verrà mostrato più avanti in questo paragrafo.

```
10 PRINT"▯"
20 INPUT R
25 PRINT"▯"
30 FOR J = 0 TO 2*π STEP .3
40 FOR K = 1 TO 10 - R*COS(J)
50 PRINT""
60 NEXT K
80 PRINTTAB(11+R*SIN(J))"●";
85 PRINT"▮"
90 NEXT J
```

Listato 10.2 - Questo programma disegna cerchi di raggi arbitrari.

Il Listato 10.2 mostra un programma che stampa un cerchio (con i simboli). Siccome i caratteri del video hanno una forma rettangolare, esso raffigura un'ellisse più che un vero cerchio. Ad ogni modo, illustra il metodo necessario per disegnare cerchi, che può essere utile in un certo tipo di programmi, specialmente in quelli che hanno a che fare con navicelle spaziali orbitanti o satelliti. Mostra inoltre come possono essere calcolati in un programma i dati relativi a tali oggetti, anche se non viene utilizzato come base per un display. Questo programma vi permette di introdurre il raggio desiderato (da 0 a 10, non necessariamente intero).

Prima di procedere esaminando i calcoli vi sono alcune cose da osservare nella linea 30. Il carattere 'π' viene battuto usando il tasto di SHIFT con quello della freccia verso l'alto. La particolarità è che non si tratta di un simbolo speciale come i cuori, fiori o altri caratteri grafici che troviamo sui tasti. Si tratta di un numero molto ben preciso, 3,14159265, che rapporta il raggio di un cerchio con la lunghezza della sua circonferenza. Si verifica che 2 π radianti corrispondono a 360° e così, se J viene fatto variare da 0 a $2^*\pi$, rappresenta un angolo crescente da 0° a 360°, un giro completo. Questo semplifica il lavoro, in quanto non dobbiamo convertire i gradi in radianti prima di calcolare i seni e i coseni nelle linee da 40 ad 80.

La figura 10.1 mostra come calcoliamo dove raffigurare i caratteri gra-

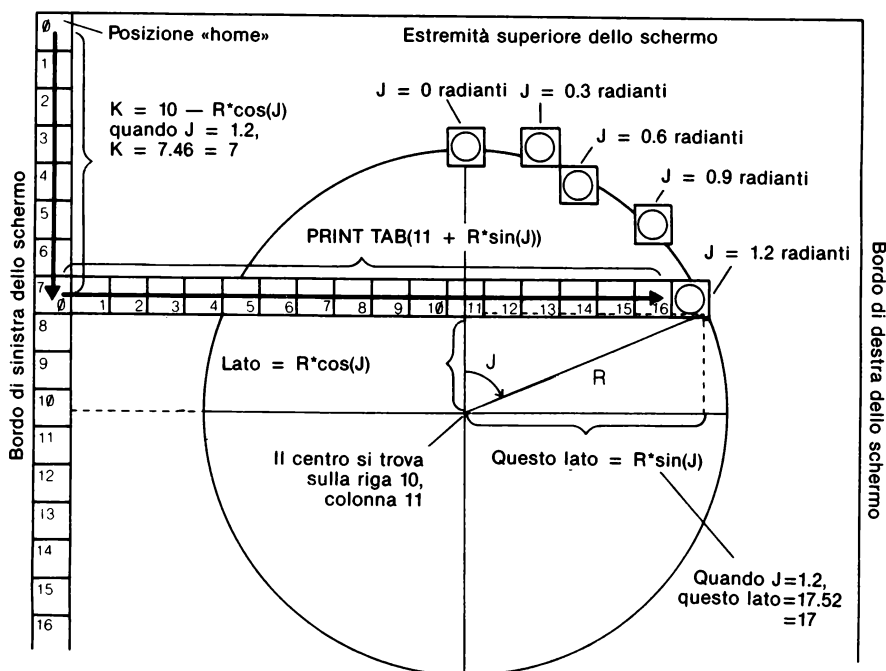


Fig 10.1 - Funzionamento del programma per disegnare i cerchi. Le freccette mostrano il moto del cursore prima che venga disegnato un cerchio.

fici tipo disco che formano la circonferenza. Il centro del cerchio viene posto sulla decima riga, undicesima colonna dello schermo. Per disegnare ciascun disco, il cursore deve muoversi per lo schermo in verticale e orizzontale per un certo numero di righe e colonne. Poi il disco è realizzato. Il numero di righe in verticale viene determinato dal loop nelle linee da 40 a 60. Il loop gira K volte ed ogni volta stampa ' ', facendo scendere il cursore di una linea. Il valore massimo di K per ciascun punto della circonferenza è calcolato da:

10 — R*COS(J)

La figura 10.1 mostra come è stata ricavata questa espressione. Una volta arrivato alla riga giusta, il cursore si muove lungo la riga usando la funzione TAB. La distanza da percorrere è calcolata da:

11 + R*SIN(J)

I valori calcolati nell'espressione precedente sono esatti e possono essere memorizzati per utilizzi successivi. In un programma che simula un volo

in orbita, essi danno le coordinate spaziali dell'astronave con una precisione di nove cifre. Ma prima di essere utilizzati per disegnare i cerchi in questo programma, vengono arrotondati al più vicino numero intero. Non è infatti possibile per il VIC disegnare un disco esattamente sopra alla riga numero 4.53458761 ed alla colonna numero 13.6543874! Con il numero limitato di righe e colonne del VIC, il cerchio che si ottiene è inevitabilmente approssimativo.

Sempre sui cerchi, il Listato 10.3 mostra un programma che vi dice tutte le informazioni necessarie su cerchi e sfere una volta assegnatone il raggio.

```
10 PRINT"♫"
20 INPUT "RAGGIO";R
30 PRINT"CIRCONFERENZA =";2*π*R
40 PRINT"AREA =";π*R↑2
50 PRINT"VOLUME DELLA SFERA =";4*π*R↑3/3
60 PRINT"AREA DELLA SFERA =";4*π*R↑2
```

Listato 10.3 - Utilizzare a pieno 'π'.

Questo fa uso del valore 'π' in un certo numero di formule. Come potete verificare facendo girare il programma, il valore di 'π' non deve essere dichiarato all'inizio. Si tratta di un valore che è già stato fissato col computer. Per mostrarvi che 'π' può anche fungere da carattere normale, abbiamo chiamato questo programma 'Uso di π'.

Un altro uso di 'π' viene fatto per le conversioni tra gradi e radianti. Siccome 180° valgono esattamente radianti, la conversione è semplice:

```
10 INPUT'GRADI';D
20 PRINT 'UGUALE';D*π/180;'RADIANTI'
```

O l'operazione inversa:

```
10 INPUT'RADIANTI';R
20 PRINT'UGUALE';R*180/π;'GRADI'
```

Le espressioni precedenti possono essere utilizzate in qualsiasi programma che richieda quelle trasformazioni.

La funzione function

Come vi è stato spiegato nel paragrafo precedente, una funzione BASIC prende un numero e ne 'fa qualcosa'. Il BASIC del VIC ha un assorti-

mento di funzioni di vario genere che possono esservi utili. In questo libro abbiamo già usato, in un modo o nell'altro, buona parte di essi.

Vi può capitare di avere bisogno di una funzione che non è compresa tra quelle fornite dal VIC. In questo caso siete liberi di scriverla voi stessi. Per esempio potrebbe servirvi una funzione che calcola il volume di una sfera, assegnato il suo raggio. Se vi serve solo una volta o due nel programma non vi è nessun problema nell'usare l'espressione 'V=4*π*R↑3/3' laddove dovete calcolare il valore del volume. Ma se questo calcolo deve essere ripetuto più volte, è utile dire al computer una volta per tutte come calcolare il volume. Poi basta semplicemente assegnare ogni volta il raggio della sfera ed utilizzare l'informazione memorizzata su come fare il calcolo. Questo è l'operato della funzione function.

Prima che una funzione definita dall'operatore possa essere utilizzata deve essere completamente specificata. Esiste a questo scopo l'istruzione DEF (definizione). Questa viene posta su una linea del programma e, proprio perchè la function deve essere definita prima di essere usata, è meglio mettere questa linea all'inizio del programma:

```
10 DEF FNV(R) = 4*π*R ↑ 3/3
20 INPUT R
30 PRINT FNV(R)
```

La linea 10 definisce la funzione V (come volume) che calcola il volume della sfera assegnato il raggio R. Il nome di questa function è V, ma avrebbe potuto avere qualsiasi altro nome del tipo che si assegna alle variabili. Generalmente si usano una o due lettere, o una lettera ed un simbolo, ma andrebbe benissimo anche FNVOLUME, o addirittura FNJIM! L'argomento della funzione (R) è il valore col quale essa deve 'fare qualcosa'.

Nell'esempio precedente la variabile R è stata usata sia come argomento della funzione che nell'INPUT di un valore di prova per richiamare la funzione nella linea 30. Non vi è la necessità di utilizzare lo stesso nome della variabile. Quando viene definita una function, la variabile che viene utilizzata nell'argomento deve, ovviamente, essere anche usata nella definizione. Quello che stiamo dicendo alla macchina effettivamente è: 'Prendi un valore, chiamalo R, elevalo al cubo, moltiplica per 4 e per 'pi', dividi per 3 e scrivi il risultato'. Mentre usiamo questa funzione, però, la variabile sulla quale essa opera può avere qualsiasi nome. Potreste cambiare le linee 20 e 30 del programma in:

```
20 INPUT T
30 PRINT FNV(T)
```

Il computer allora prenderà il valore di T e ne farà quello che gli era stato

detto di fare con R. Nell'argomento potete anche avere un'espressione numerica. Per esempio, cambiate la linea 30 in:

```
30 PRINT FNV(2*T+7)
```

Questo fa sì che T venga raddoppiato, gli si aggiunga 7 e, solo ora, lo si mandi alla function. Il risultato è il volume di una sfera con raggio pari al doppio di T aumentato di 7.

Funzioni di funzione

Quando avete fatto girare il programma Uso di π, vi potete essere mervigliati di tutte le cifre che calcola. Vi dice, per esempio che una sfera di raggio 4 cm. ha un volume di 268.082573 cm^3. Una delle cause di una risposta così lunga è il fatto che il valore usato per π ha nove cifre significative. All'atto pratico dei numeri con tante cifre sono quasi senza senso. Se il raggio della sfera viene calcolato con precisione fino al millimetro (0.1 cm), non ha senso calcolare il volume con una precisione di un milionesimo di centimetro cubo. È molto meglio arrotondare la risposta al più vicino numero intero. La funzione INT non fa esattamente questo, in quanto essa toglie tutte le cifre dopo la virgola. INT(5.9) vale 5. Se arrotondassimo 5.9 nel modo abituale, avremmo logicamente 6. Questo perchè i numeri al di sotto della metà dell'intero successivo (ad esempio meno di 5.5) vengono arrotondati per difetto (a 5), mentre quelli dalla metà in su (maggiori o uguali di 5.5) vengono arrotondati per eccesso (a 6).

Arrotondare al più vicino numero intero è semplice. Se infatti aggiungiamo 0.5 al numero da arrotondare prima di utilizzare INT, il numero viene arrotondato correttamente:

```
10 INPUT N
20 PRINT INT(N+.5)
```

Provate questo programma con più numeri, come ad esempio 5.3, 5.5, 5.7, —3.234, o qualsiasi altro. L'arrotondamento è sempre corretto, anche coi numeri negativi. Possiamo farlo diventare una function R (per rounding, arrotondare), che viene definita come segue:

```
10 DEF FNR(N)=INT(N+.5)
```

Possiamo ora aggiungere questa function al programma 'Uso di π', per fargli visualizzare risultati arrotondati al più vicino numero intero (vedere il Listato 10.4).

```
5 DEF FNR(N) = INT(N+.5)
10 PRINT"□"
20 INPUT "RAGGIO";R
30 PRINT"CIRCONFERENZA =";FNR(2*π*R)
40 PRINT"AREA =";FNR(π*R↑2)
50 PRINT"VOLUME DELLA SFERA =";FNR(4*π*R↑3/3)
60 PRINT"AREA DELLA SFERA =";FNR(4*π*R↑2)
```

Listato 10.4 - Come definire ed utilizzare una function.

La nuova versione del programma contiene nella linea 5 la definizione della funzione di arrotondamento. Il resto del programma è lo stesso di prima, tranne che le espressioni usate nei calcoli sono a loro volta soggette ad arrotondamento.

Veniamo ora alle funzioni di funzioni! Se noi abbiamo la funzione V, per cercare il volume di una sfera, e la funzione R per arrotondare il risultato, possiamo combinare le due cose? Provate il Listato 10.5. La linea 10 definisce la funzione di arrotondamento R, mentre la linea 20 definisce la funzione V per calcolare il volume della sfera, assegnatone il raggio.

```
10 DEF FNR(N) = INT(N+.5)
20 DEF FNV(R) = 4*π*R↑3/3
30 INPUT R
40 PRINT FNV(R)
50 PRINT FNR(FNV(R))
```

Listato 10.5 - Una funzione di funzione!

Nella linea 40 il volume viene stampato con nove cifre significative. Nella linea 50 la funzione R della funzione V arrotonda la risposta al più vicino numero intero. Così, nonostante sembri una cosa astratta, la funzione di funzione ha i suoi utilizzi pratici.

Prima di lasciare la funzione di arrotondamento, eccovi le funzioni per arrotondare le altre cifre decimali:

Per arrotondare fino alla prima cifra:

INT((N+.05)*10)/10

Per arrotondare fino a 2 cifre decimali:

INT((N+.005)*100)/100

Per arrotondare fino a 3 cifre decimali:

INT((N+0.0005)*1000)/1000

Definire una o più di queste funzioni può preservare molto spazio in memoria, qualora l'arrotondamento venga fatto più volte nel programma.

I numeri casuali sono spesso necessari nei programmi dei giochi e sono utili per dare, con un minimo di programmazione, vari colori ai display grafici. Il programma del tappeto magico (Listato 3.3) fa uso di cinque diversi numeri casuali per ottenere i suoi effetti. Se pensiamo di utilizzare i numeri casuali in più punti del programma e se, come è da auspicarsi, questi devono essere estratti da campi diversi, una function 'numeri casuali' può farci risparmiare molto spazio. Provate il semplice programma nel Listato 10.6.

```
10 DEF FNR(N) = INT(RND(1)*(N+1))
20 INPUT N
30 FOR J = 1 TO 100
40 PRINT FNR(N);
50 NEXT J
```

Listato 10.6 - Una funzione pratica per generare numeri casuali.

Questo programma assume un numero N e poi stampa 100 numeri nell'intervallo da 0 a N. Potete usare questa funzione a qualsiasi livello di qualsiasi programma, per generare un numero casuale compreso in un qualunque intervallo da zero ad N. Basta utilizzare una linea simile alla linea 40 del programma precedente, sostituendo al posto di N un qualunque numero, variabile o anche espressione numerica. Se volete che questo numero sia nell'intervallo da 1 a N, invece che da 0 a N, utilizzate questa definizione di funzione:

10 DEF FNR(N)=INT(RND(1)*N)+1

Introducendo il valore 6 in questo programma ottenete la simulazione di un dado con le facce numerate nel modo abituale, da 1 a 6.

Riassunto

In questo capitolo avete visto come:

— utilizzare i 4 tasti funzionali nei programmi con interazioni esterne
— utilizzare la funzione POS per determinare dove deve essere stampato il carattere successivo
— utilizzare la funzione SQR per trovare la radice quadrata di un numero
— rappresentare un cerchio sullo schermo
— calcolare le coordinate di un oggetto che si muove su un'orbita circolare
— convertire gradi in radianti e radianti in gradi

— definire funzioni di vostra personale utilità ed usarle nei programmi
— arrotondare numeri ad ogni grado di precisione.

Avete imparato che:

— 'funzione' è una parola abusata con diversi significati distinti
— π è sia un simbolo che può essere visualizzato sullo schermo che un numero con un valore ben determinato.

Capitolo 11
Colori e movimento

In alcuni dei capitoli precedenti abbiamo visto vari programmi per generare immagini che si muovono sullo schermo. Ora ci spingeremo oltre nella conoscenza della tecnica della grafica animata del VIC. I caratteri speciali utilizzati in questo capitolo si ottengono tutti utilizzando i programmi ausiliari Newkey e Multikey descritti nel capitolo sette (Listati 7.3 e 7.4). Se pensate di scrivervi dei programmi personali di grafica e movimento, vi conviene introdurre questi programmi ausiliari e registrarli su nastro.

In questo capitolo vengono descritte diverse maniere per generare immagini in movimento, ciascuna con un esempio. In ognuna di esse l'operazione fondamentale consiste nel rimpiazzare un'immagine con una leggermente differente. Se ciò viene fatto in maniera sufficientemente veloce, si realizza una illusione di movimento. Talvolta si sposta l'intero oggetto, visualizzandolo in righe o colonne successive. Tutta l'immagine sembra così muoversi sullo schermo. Possiamo anche muovere solo una parte di oggetto, visualizzando un'immagine leggermente diversa sulla stessa zona dello schermo. L'oggetto sembra così fermo, ma pare muovere alcune sue parti. Questi due tipi di movimento possono essere combinati, così che l'oggetto sembra muoversi lungo lo schermo muovendo anche parte di sè. In questo capitolo troverete esempi di tutti questi modi di animazione.

Programmare la grafica dei movimenti

Questo paragrafo serve per scrivere molti dei programmi descritti in questo capitolo, ma non tutti. Questi programmi possono essere utilizzati in tre modi diversi. Tutti questi utilizzano le stesse routines per visualizzare e far muovere le immagini sullo schermo.

Se avete già usato Newkey o Multikey (vedere il punto (1) qui sotto), potete già visualizzare i caratteri e scrivere con essi piccoli programmi. Questo si verifica finchè battete RUN/STOP con il RESTORE o spegne-

te la macchina. Una procedura leggermente differente è necessaria quando volete usare nuovamente questi caratteri nel vostro programma. Vi sono due modi per fare ciò (vedere i punti (2) e (3) sotto) e la scelta dell'uno o dell'altro dipende da quanta memoria vi serve per il vostro programma.

(1) Dopo Newkey o Multikey

Cominciate a caricare uno di questi due programmi ausiliari e fatelo girare. Ciascun esempio in questo capitolo ha un disegno che mostra i caratteri richiesti. Usate questi disegni per scegliere i caratteri da utilizzare. Questi sono memorizzati sotto i codici definiti a fianco di ogni disegno. Quando avete finito di scegliere tutti i caratteri per il vostro programma, controllate che gli otto codici di ciascun carattere coincidano con quelli definiti a fianco dei disegni.

Newkey e Multikey contengono entrambi una serie di indirizzi che fanno accettare al VIC le vostre nuove definizioni al posto dei caratteri normalmente associati a quei codici. Essi copiano inoltre nella RAM i caratteri della parte superiore e del lato di destra, così voi potete continuare ad usarli. Il VIC conserva queste disposizioni a patto che voi non battiate RUN/STOP con RESTORE (o lo spegniate), anche se cancellate Newkey o Multikey, battendo NEW. Potete inoltre modificare questi listati o scrivere programmi completamente nuovi che fanno uso degli stessi caratteri.

Ad ogni modo, se memorizzate il programma e poi spegnete, questo non funzionerà una volta caricato di nuovo, in quanto il VIC farà uso del suo generatore di caratteri e visualizzerà delle X, Y o Z al posto dei caratteri speciali. La soluzione è l'uso del metodo (2) qui riportato.

(2) Programmi indipendenti

Tutto quello che c'è da fare per avere un listato indipendente delle utility, è aggiungere poche linee supplementari che tengano conto delle funzioni essenziali dell'utility (vedere il Listato 11.1).

```
1 POKE 51,255:POKE 52,23:POKE 55,255
2 POKE 56,23:POKE 36869,254
3 FOR J= 0 TO 1023
4 X = PEEK(32768 + J):POKE 6144 + J,X
5 NEXT J
6 FOR J = 0 TO 31
7 READ X
8 POKE 6328 + J,X:NEXT J
```

Listato 11.1 - Introduzione dei simboli grafici definiti dall'utente nei programmi: come iniziare il programma.

A queste vengono assegnate delle etichette in modo che possano essere inserite all'inizio dei listati, che cominciano tutti con la linea 10. La linea 1 regola l'inizio della memoria e la lunghezza della stringa, facendo poi

prendere al computer i suoi caratteri dalla RAM (da 6144 in su) invece che dal suo generatore di caratteri. Le linee da 2 a 4 riportano i codici delle lettere maiuscole e la grafica del lato destro alla RAM dal generatore di caratteri ROM.

Le linee da 5 ad 8 riportano alla RAM i codici di ogni carattere speciale (presi dalle istruzioni DATA) rimpiazzando i codici delle lettere maiuscole. Le istruzioni DATA vengono inserite come linee aggiuntive alla fine di ogni programma. Le cifre da inserire nelle istruzioni DATA sono quelle stampate in fianco ad ogni carattere speciale. Per esempio nel Listato 11.2 vengono mostrate le istruzioni DATA per il programma del Talker.

```
1000 DATA 126,255,187,153,255,231,231,12
6,66,126,0,0,0,0,0,0
1010 DATA 126,255,221,153,255,231,231,12
6,66,102,24,0,0,0,0,0
```

Listato 11.2 - Una tipica istruzione DATA, da inserire alla fine del programma.

I valori assegnati nelle routine qui sopra, vi permettono di avere circa 2 kilobyte di RAM a disposizione del vostro programma ed avete così la possibilità di usare le lettere maiuscole e la grafica di destra in modo normale. Le cifre delle linee 5 e 7 sono corrette per un programma tipo quello del Talker (Listato 11.7), che fa uso di quattro caratteri speciali. Queste cifre devono essere cambiate, come vi viene mostrato nella tabella 12, quando il programma usa un numero diverso di caratteri.

Lo svantaggio principale di questo metodo è che impiega 25 secondi per ricopiare tutti i caratteri della tastiera. Se volete lavorare solo con i caratteri speciali e non vi interessa avere a disposizione anche le lettere maiuscole o i caratteri dei tasti di destra, non vi è motivo di perdere tempo trasportando tutti questi nella RAM. Potete sempre battere RUN/STOP con il RESTORE per riaverli temporaneamente, per scrivere il programma. Per risparmiare il tempo di trascrizione, usate le linee da 2 a 4 del Listato 11.3 al posto di quelle date prima nel Listato 11.1.

```
2 FOR J = 0 TO 7
3 POKE 6400 + J,0
4 NEXT J
```

Listato 11.3 - Linee alternative a quelle del Listato 11.1, che trasferiscono solo lo 'spazio' alla RAM.

Le linee trasferiscono solo uno dei caratteri normali, lo spazio. Questo è essenziale, in quanto senza di esso lo schermo appare a macchie invece di avere un colore uniforme.

(3) Programmi indipendenti con massima memoria per programmare

Questa è una modifica del precedente metodo (2), ma pone più tardi i

```
60 PRINT"▮NOI ♥ IL VIC!"
```

Listato 11.4 - Una tipica istruzione di PRINT da usarsi col metodo (3).

caratteri nella RAM. Lascia circa 3 kilobytes per i vostri programmi. I tasti stampano lettere maiuscole, (ad eccezione di quelli che sono appena stati ridefiniti come caratteri speciali) ma non vi sono caratteri grafici. Se volete usarne uno di quelli del set di destra, è facile recuperarli. Cominciate ogni istruzione di PRINT che contiene uno di questi caratteri con un comando 'invertito', come mostrato nel Listato 11.4. L''inversione' si ottiene battendo CTRL col tasto 9. I simboli dei cuori, fiori o altri si ottengono alla solita maniera, battendo SHIFT con il tasto opportuno. Non vedrete alcun cuori, fiori o altri simboli riconoscibili mentre state introducendo una linea del programma; appare invece una strana combinazione di punti! Non vi preoccupate, poichè tutto è a posto. Al momento dell'esecuzione della linea, i cuori o gli altri simboli appariranno sullo schermo come voi avevate previsto. Essi appariranno correttamente anche nei listati ottenuti con la stampante VIC 1515. Potete includere coi simboli le lettere maiuscole; queste appariranno normalmente sia quando le battete, sia sullo schermo una volta visualizzate.

```
1 POKE 51,255:POKE 52,27:POKE 55,255
2 POKE 56,27:POKE 36869,255
3 FOR J = 0 TO 511
4 X = PEEK(32768 + J):POKE 7168 + J,X
5 NEXT J
6 FOR J = 0 TO 31
7 READ X
8 POKE 7352 + J,X:NEXT J
```

Listato 11.5 - Questa versione del Listato 11.1 utilizza meno memoria.

Il Listato 11.5 vi dà le linee da scrivere all'inizio di tutti i programmi in cui volete avere il massimo di memoria. Sono quasi le stesse usate per il metodo (2), ad eccezione di alcuni valori. Come per il metodo (2), il programma impiega parecchi secondi per trascrivere le lettere maiuscole sulla RAM. Fortunatamente ci mette solo la metà del tempo impiegato dal metodo (2) in quanto non deve trascrivere i simboli di destra. Se volete eliminare questo passaggio, trascrivete solo lo spazio, sostituendo le linee nel Listato 11.5.

I valori delle linee da 5 a 7 devono essere utilizzati solo quando i caratteri definiti dall'utente sono quattro. Nel caso di un diverso numero di

```
2 FOR J = 0 TO 7
3 POKE 7424 + J,0
4 NEXT J
```

Listato 11.6 - Fate uso di queste linee, nel metodo (3), per iniziare il vostro programma.

caratteri dovete usare i valori dati dalla tabella 13. Come nel metodo (2), le caratteristiche dettagliate dei nuovi caratteri vengono introdotte come linee DATA del programma. Queste linee sono esattamente le stesse usate nel metodo precedente.

La programmazione di grafici a colori può sembrare cosa difficile, ma è un'impressione dovuta al fatto che vengono presentati più metodi alternativi. È meglio che vi soffermiate su uno di questi fino a che non avrete familiarizzato con esso. Se siete dei principianti, conviene cominciare col metodo (1). Tutto quello che dovete fare è caricare Newkey o Multikey (Listati 7.3 e 7.4) e definire i caratteri. Vi divertirete certamente ad osservarne i risultati. Quando poi avrete fatto un po' di pratica e vorrete scrivervi dei programmi vostri, provate il metodo (2) o (3), magari usando per primo il metodo (2) in quanto si tratta del più conveniente. Quando i vostri programmi tenderanno a diventare sempre più lunghi, vi converrà passare al metodo (3).

Parti in movimento

Il programma del Talker (Listato 11.7) mostra una testa che apre e chiude la bocca, girando nello stesso tempo da parte a parte gli occhi. La figura è ottenuta usando due coppie di caratteri grafici (fig.11.1). A queste viene fatto riferimento con le lettere dei codici al posto dei quali vengono memorizzate. Due di questi caratteri vengono stampati l'uno sull'altro al centro dello schermo. Quando viene stampato W su X, vediamo una testa con

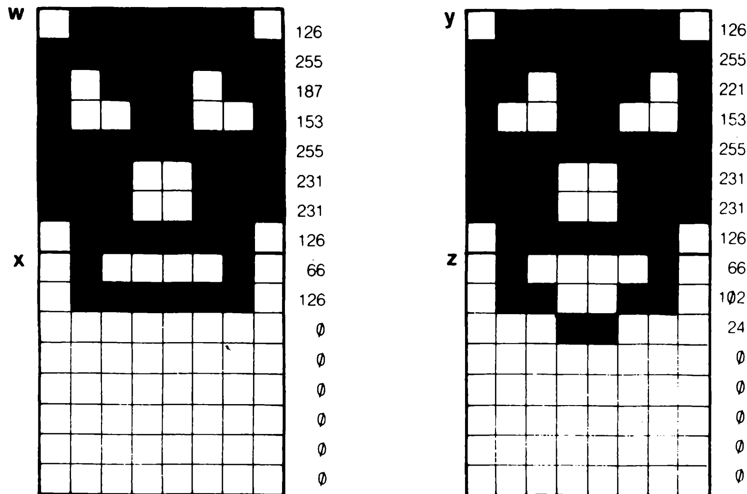


Fig. 11.1 - I caratteri usati nel programma del Talker.

la bocca chiusa e gli occhi rivolti verso la destra dello schermo. Quando Y è stampato sopra Z, vediamo la testa con la bocca aperta e gli occhi che guardano verso sinistra. Se W su X e poi Y su Z vengono visualizzati alternatamente in rapida successione, sembra che la testa stia parlando, ruotando gli occhi.

```
10 PRINT"□■":POKE 36879,8
20 PRINT"QQQQQ"
30 PRINTTAB(10)"W"
40 PRINTTAB(10)"X"
50 FOR J = 1 TO 200:NEXT J
60 PRINT"S"
65 PRINT"QQQQQ"
70 PRINT"QQQQQ"
80 PRINTTAB(10)"Y"
90 PRINTTAB(10)"Z"
100 FOR J = 1 TO 200:NEXT J
110 PRINT"S":GOTO 20
```

Listato 11.7 - Il Talker.

La linea 10 fissa bianco il colore da stampare e nero sia il bordo che lo schermo, ottenendo così l'effetto della testa desiderato. Le linee da 20 a 70 fanno spostare il cursore di cinque file verso il basso prima di stampare i caratteri. Per battere questa linea, introducete prima PRINT", poi battete cinque volte il tasto (senza lo shift) di spostamento del cursore alto-basso. Poi battete ". Le linee 60 e 110 rimandano all'inizio il cursore senza cancellare lo schermo (CLR/HOME senza shift). Nelle linee 30, 40, 80 e 90 potete vedere che i caratteri compaiono nel listato come W, X, Y e Z. Sul video, però, compariranno come una testa che parla.

Moto e locomozione

La figura 11.2 mostra i disegni da realizzare per dare l'illusione di vedere un uomo che cammina per lo schermo (Walker). Vi è una coppia di caratteri, detti U e V, ed una seconda coppia, W e Z. U e V rappresentano una persona che ha appena fatto un passo avanti. Sarebbe possibile visualizzare sullo schermo solo U e V in colonne successive, ma il risultato sarebbe scadente. Funzionerebbe meglio col Puffer (Listato 7.1) perchè con la scala con la quale è stata disegnata la locomotiva, non ci si aspetta di vedere le ruote muoversi e, di conseguenza, i suoi pistoni. Le sue ruote infatti non hanno raggi ed i pistoni non sono stati disegnati, così che non vi sarebbe niente che disturba l'effetto. La persona, invece, ha le gambe che sono perfettamente visibili e che ci debbono dare l'immagine del camminare. Se rappresentassimo solamente U e V su colonne successive, la persona sembrerebbe scivolare sullo schermo come sul ghiaccio.

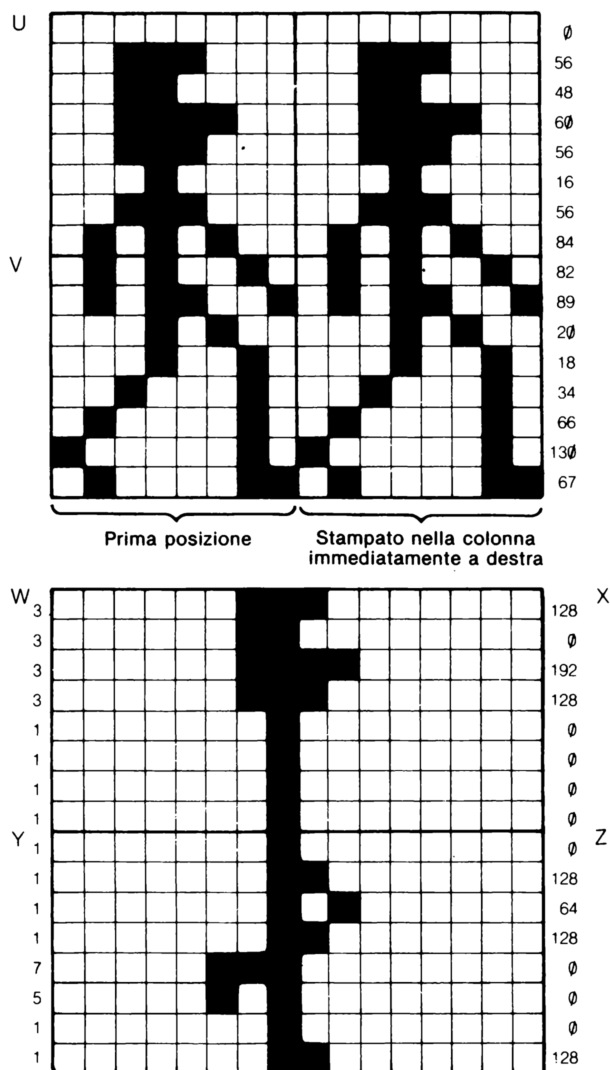


Fig.11.2 - I caratteri usati nel programma del Walker.

W su Z collega due visualizzazioni successive di U su V. La persona sta camminando a cavallo di due colonne successive dello schermo ed appare in parte nella prima, in parte nella seconda colonna. Questo è il motivo per cui abbiamo dovuto utilizzare quattro caratteri. Paragonando le due coppie di caratteri, vediamo che nel secondo il piede avanti è leggermente avanzato, come se si fosse effettivamente appoggiato a terra. Il piede posteriore è stato portato in avanti e la gamba è piegata all'altezza del ginocchio come accade nella realtà. Le braccia non si vedono perchè cor-

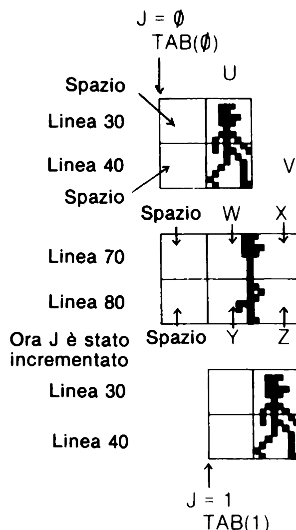


Fig. 11.3 - In questo modo viene fatta apparire sullo schermo la figura che cammina.

rono lungo il corpo, oscillando l'una in avanti, l'altra indietro. La testa si alza in quanto la persona è allo stadio perfettamente verticale della camminata.

L'ordine di stampa dei caratteri è mostrato in fig. 11.3. Vengono visualizzati come stringhe. Queste comprendono degli spazi per coprire i caratteri precedentemente visualizzati. Gli spazi stampati prima di W/X e Y/Z non sono rigorosamente necessari poichè W e Y si stampano sopra ad U e V. Ad ogni modo, se tralasciamo questi spazi, dobbiamo modificare i TAB delle linee 70 ed 80 in TAB(J+1), rendendo più facile l'inclusione di spazi addizionali. Il programma viene mostrato nel Listato 11.8.

```
10 PRINT"[illegible]"
20 FOR J = 0 TO 18
30 PRINTTAB(J)" U"
40 PRINTTAB(J)" V"
50 PRINT"[illegible]"
60 FOR K = 1 TO 100:NEXT K
70 PRINTTAB(J)" WX"
80 PRINTTAB(J)" YZ"
90 PRINT"[illegible]"
100 FOR K = 1 TO 100:NEXT K
110 NEXT J
```

Listato 11.8 - Il Walker.

Stampare o inizializzare?

Stampare i caratteri sullo schermo ha il vantaggio di semplificare la fase di scrittura ed è generalmente più facile per eventuali altri utenti leggere

e comprendere il programma. Uno svantaggio è che la visualizzazione può interferire con altre parti del display. Inizializzare i caratteri permette di collocarli dove è necessario senza interferire con alcun indirizzo del video. Inoltre è più facile spostarli sullo schermo usando dei loop che contengono istruzioni di POKE e variabili piuttosto che fare simili operazioni con istruzioni di PRINT.

```
10 PRINT"♥":D = 30720
20 FOR J = 7724 TO 7742
30 POKE J+D,1:POKE J+D+1,2:POKE J+1,21
40 POKE J+D+22,1:POKE J+D+23,2:POKE J+23,22
60 FOR K = 1 TO 100:NEXT K
70 POKE J+D+2,2:POKE J+1,23:POKE J+2,24
80 POKE J+D+24,2:POKE J+23,25:POKE J+24,26
100 FOR K = 1 TO 100:NEXT K
110 NEXT J
```

Listato 11.9 - Questa versione del Walker sfrutta le istruzioni di POKE al posto delle PRINT.

Se confrontate questo con il Listato 11.8, è facile scoprire le similitudini tra i due. Viene assegnato a W il valore del primo indirizzo dello schermo sul quale deve essere introdotta la metà superiore della persona. La variabile D rappresenta la differenza tra gli indirizzi del codice RAM dei caratteri e del codice RAM dei colori (vedere la fig. 3.3). Ogni comando contenente D determina il colore col quale verrà visualizzato il carattere corrispondente.

I comandi della linea 30 fanno quanto segue:

POKE J + D,1	Visualizza in bianco; siccome lo sfondo è bianco, questo comando pulisce lo schermo da un carattere precedentemente visualizzato. Si tratta dell'equivalente della visualizzazione di uno spazio.
POKE J+D+1, 2	Visualizza in rosso; il programma mostra una persona rossa.
POKE J + 1,10	Fa comparire la metà superiore della persona nell'indirizzo dello schermo J. La metà superiore viene specificata sotto il codice 'U', quindi per trovare il numero da introdurre dovete guardare la tabella 6 dell'Appendice A. Lì troverete che il codice video di 'U' è '21'.

I comandi nella linea 40 sono simili, ma si riferiscono alla metà inferiore della persona. Gli indirizzi da introdurre sono tutti quelli contenuti nella linea 30 aumentati di 22 per fare in modo che la metà inferiore appaia esattamente sotto a quella superiore. Il codice video di 'V' è '22'. Nelle

linee 70 ed 80 i caratteri W e Z vengono introdotti nelle loro rispettive posizioni al posto di U e V, che vengono resi invisibili facendoli rappresentare in bianco.

Collisioni

Se le persone e gli oggetti devono muoversi sullo schermo in modo realistico, devono comportarsi come effettivamente fanno persone e cose. In particolare, quando vengono in contatto con altre persone o cose, devono fermarsi, o rimbalzare o comportarsi in qualche modo come se fosse avvenuta un'effettiva collisione. Questo è essenziale nei programmi dei giochi, nei quali vogliamo che una palla rimbalzi su di un muro, o che un battitore colpisca una pallina. Possiamo anche richiedere di vedere quando un missile colpisce il suo bersaglio. Nel gioco di Cupido (Listato 11.10), i missili sono le frecce di Cupido ed i bersagli, logicamente, sono i cuori.

```
10 PRINT"[illegible]":POKE 36879,205:R = 8164
20 FOR J = 1 TO 20
30 PRINTSPC(INT(RND(1)*20));"♥"
40 NEXT
50 GET A$
60 IF A$ <> CHR$(133) THEN GOTO 50
70 FOR J = 0 TO 65
80 GET A$:IF A$ = CHR$(136) THEN R = R - 22
90 IF PEEK(R+J)=83 THEN GOSUB 1000
100 POKE R + J,30:POKE R+30720+J,5
110 FOR K = 1 TO 500:NEXT K
120 NEXT J
130 PRINT"[illegible]";TAB(2)"TU SEI A";N;"DA TERRA"
140 GOTO 140
1000 N = N + 1:POKE 36878,15:T = 150
1010 FOR L = 1 TO 50
1020 T = T - 6
1030 POKE 36876,T
1040 FOR M = 1 TO 10:NEXT M
1050 T = T + 8
1060 POKE 36876,T
1070 FOR M = 1 TO 10:NEXT M
1080 NEXT L
1090 POKE 36876,0:POKE 36878,0
1100 RETURN
```

Listato 11.10 - Cupido, un gioco che mostra come controllare le collisioni.

Siccome il programma fa uso dei caratteri grafici del VIC e non richiede all'utente di definirne dei propri, i metodi descritti nel paragrafo precedente non servono. Vi basta quindi introdurre il programma e farlo girare.

Prima di discutere delle collisioni tra frecce e cuori, rivolgiamo la nostra attenzione ad una nuova funzione che compare alla linea 30. SPC(N) viene utilizzata nelle istruzioni di PRINT per fare visualizzare N spazi.

Nella linea 30 l'argomento di SPC è un numero casuale compreso tra zero e 19. L'effetto quindi delle linee da 20 a 40 è quello di visualizzare sullo schermo 20 cuori. La posizione dei cuori sullo schermo è diversa ogni volta che viene fatto girare il programma. L'istruzione di GET A$ nella linea 50 dà all'utente il tempo di osservare la disposizione sullo schermo e di preparare una tattica. Quando il giocatore batte il tasto funzionale F1 il gioco comincia.

Il loop dalla linea 70 alla linea 120 fa apparire una serie di frecce verdi rivolte verso l'alto sullo schermo. La visualizzazione avviene sempre all'indirizzo R ed R è inizializzato a 8164 (linea 10) così che la prima freccia appare nell'angolo in basso a sinistra. Durante il loop R viene di volta in volta incrementato, le frecce appaiono di conseguenza sullo schermo in colonne successive e questo viene così incorniciato tre volte. Quando il giocatore batte il tasto F7 (linea 80), R viene diminuito di 22; facendo visualizzare la freccia sulla linea immediatamente sopra. In questo modo la fila di frecce può essere orizzontale (quando F7 non viene battuto) o inclinata in alto a destra (se F7 viene battuto).

Le collisioni vengono controllate nella linea 90 leggendo con il PEEK gli indirizzi (ad R+J) nei quali deve essere rappresentata la freccia successiva. Se si verifica che PEEK(R+J) vale '83', che è il codice di schermo del 'cuore', la collisione è inevitabile, in quanto nella linea successiva del programma nello stesso indirizzo verrà introdotto il codice di schermo di una freccia. Prima di colpire il bersaglio, il programma passa comunque ad una subroutine che tiene conto dei cuori colpiti (N) e produce un effetto sonoro.

Quando il programma principale viene ripetuto 66 volte e la fila di frecce si stende per tre volte sullo schermo, il gioco finisce visualizzando il punteggio finale (linea 130). Battete RUN/STOP con RESTORE se volete giocare ancora. Ci vuole un certo allenamento per trovare la strategia ottimale in ciascuna disposizione dei cuori. Quando vi sentirete abbastanza esperti, rendete più veloce il programma riducendo la lunghezza dei loop di ritardo nella linea 110.

Colori che si muovono

Il modo convenzionale di ottenere oggetti in movimento sullo schermo è di stamparli su caratteri successivi o indirizzarli in codici successivi dello schermo. La stampa può danneggiare altre immagini già presenti sullo schermo, mentre l'indirizzamento è un po' difficile da realizzare in quanto entrambi i codici dei caratteri della RAM ed i codici dei colori delle ROM devono essere indirizzati di volta in volta. Qui vi presentiamo un modo per semplificare le procedure e velocizzare l'operazione. In primo

luogo, il codice di carattere viene introdotto in tutti gli indirizzi dello schermo per i quali deve passare l'oggetto. Quando questo sta per apparire nell'indirizzo assegnatogli, il codice dei colori della RAM viene caricato con l'indirizzo del colore desiderato (non il bianco). L'oggetto appare immediatamente! Per farlo muovere all'indirizzo successivo, il suo attuale codice del colore viene modificato in 1 con un POKE, per visualizzarlo in bianco (il che significa farlo scomparire), mentre il codice di colore del successivo indirizzo viene caricato con il codice di colore voluto. Scomparendo da un indirizzo e riapparendo nel successivo, l'oggetto dà l'illusione di muoversi.

```
10 PRINT"♥"
20 FOR J = 7724 TO 7745
30 POKE J,81
40 NEXT J
50 FOR J = 38444 TO 38465
60 POKE J-1,1:POKE J,5
70 FOR K = 1 TO 200:NEXT K
80 NEXT J
```

Listato 11.11 - Creare l'illusione del movimento semplicemente cambiando i colori dello schermo.

Il programma nel Listato 11.11 mostra questa tecnica. Questo è un altro programma da introdurre e fare girare. Le linee 20 e 30 inroducono il codice di un disco in tutti i codici di carattere delle caselle della terza fila dello schermo. Nulla appare a questo livello sullo schermo, ma tutto è ora pronto per entrare in azione. Alle linee da 50 a 70 gli indirizzi dei codici di colore della RAM della terza fila dello schermo vengono settati a '5' uno dopo l'altro per fare comparire il disco colorato di verde. Ad ogni passo la cella (J-1) è settata ad '1' per fare scomparire il disco precedente nel momento in cui arriva quello nuovo. Il risultato è di vedere una palla che si muove per lo schermo. Questo programma permette alla palla di muoversi solo lungo una linea, semplicemente indirizzando una sola cella per volta. Potete aggiungere al programma linee per mandare avanti e indietro la 'palla' quante volte volete, senza dover reindirizzare la sua traiettoria. Quello che fate è semplicemente 'spostare i colori'.

```
20 FOR J = 7724 TO 7745 STEP 2
30 POKE J,81:POKE J+1,67
```

Listato 11.12 - La medusa.

Una variante di questo programma, mostrata nel Listato 11.12, cambia le linee 20 e 30. Questo colloca in posizioni alternate un 'disco' ed un tratto orizzontale lungo la terza riga dello schermo. Ora, quando fate girare il programma, il disco sembra appiattirsi durante il movimento, pulsando come una medusa. Grazie a questa tecnica, con una valida scelta di caratteri, utilizzando sia quelli forniti dal VIC che quelli creati da voi stessi, potete riprodurre sullo schermo molti effetti interessanti.

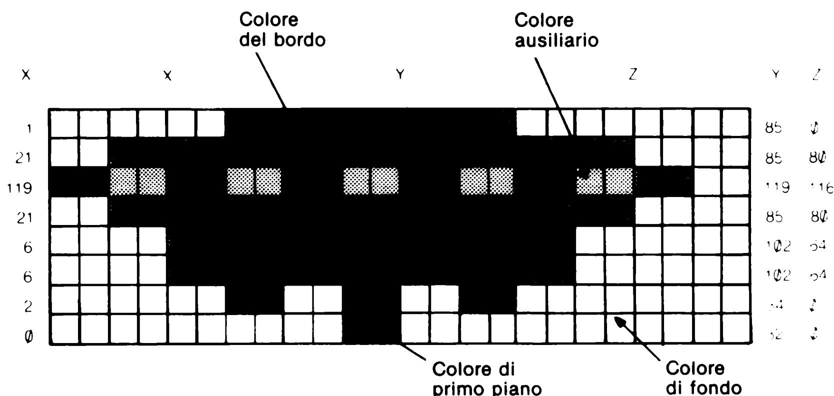


Fig. 11.4 - I caratteri usati nel programma di rappresentazione dell'UFO.

Moto di più colori

Questa tecnica sviluppa l'idea della precedente e sfrutta grafiche con più colori per migliorare gli effetti ottenibili. La figura 11.4 mostra il disegno di un disco volante come potrebbe comparire dal di fuori sul vostro schermo in qualunque istante. È formato da tre caratteri multicolori, definiti usando il programma ausiliario Multikey (Listato 1.4).

L'UFO, così lo chiameremo d'ora in poi, è nero, con luci gialle e fiamme rosso plasma provenienti dai tre razzi vettori sotto di esso. Il modo con cui i colori vengono allocati, è la chiave della tecnica. Il modo multicolore permette quattro colori. Ecco come li usiamo qui:

Colore di fondo: Questo è il colore dello schermo. In questo programma è blu, per somigliare al cielo.

Colore del bordo: Questo è il colore che usiamo per il corpo dell'UFO. Viene reso nero fissando nero il bordo dello schermo.

Colore in primo piano: Generalmente un carattere viene visualizzato col colore di primo piano, ma in questo metodo abbiamo assegnato a questo compito il colore di fondo. Il motivo sta nel fatto che il colore di primo piano può essere rapidamente cambiato introducendo gli indirizzi opportuni (in questo esempio tre) nel codice di colore della RAM. In questo programma il colore di primo piano è sia rosso, quando i razzi sono accesi, che blu (lo stesso colore del cielo), quando i razzi sono spenti.

Colore ausiliario: Questo è un altro colore che può facilmente essere cambiato, introducendo il colore richiesto nel Chip di Interfaccia all'indirizzo 36878. Quando vogliamo che le luci siano accese, poniamo in giallo il colore ausiliario. Quando invece vogliamo che le luci siano spente, mettiamo in nero il colore ausiliario, nello stesso colore cioè dell'UFO.

Riassumendo, i colori di fondo e del bordo, che non possono essere cambiati senza toccare tutto lo schermo, vengono utilizzati per le parti del disegno che si mantengono costanti. Possiamo ottenere effetti sorprendenti, cambiando i colori di primo piano e ausiliari con un minimo di programmazione. Se per di più abbiamo un'intera flotta di UFO sullo schermo, possiamo controllare contemporaneamente le luci di tutti. Il trucco sta solo nel POKE usato per cambiare il colore ausiliario. In questo programma stiamo semplicemente accendendo o spegnendo luci o razzi, ma l'idea può venire realizzata anche con il movimento di parte degli oggetti come ci mostra l'esempio del paragrafo successivo (Dancing Puppet).

Il programma nel Listato 11.13 viene battuto dopo aver definito i tre caratteri con Multikey e battendo poi NEW. I caratteri vengono assegnati ai tasti 'X', 'Y' e 'Z'.

```
10 PRINT"J":POKE 36879,104:R = 8173:N=14
20 GOSUB 1000
30 POKE 36878,112:GOSUB 2000
40 FOR K = 1 TO 100:NEXT K
50 POKE 36878,0
60 GET A$
70 IF A$ = CHR$(136) THEN N = 6:GOSUB 1000
75 R=R-22:GOSUB 2000:N = 10:GOTO 20
80 FOR K = 1 TO 100:NEXT K
90 POKE 36878,112
100 GOTO 40
1000 FOR J = 0 TO 2
1010 POKE R + 30720 + J,N
1020 NEXT J
1030 RETURN
2000 FOR J = 0 TO 2
2010 POKE R + J,24 + J
2020 NEXT J
2030 RETURN
```

Listato 11.13 - Il programma dell'UFO fa uso della grafica multicolore per far comparire su vostro comando i razzi della navicella.

La linea 10 libera lo schermo, fissa il colore di questo in blu e quello del bordo in nero. Poi fissa in N il colore di primo piano che alla fine sarà blu (a razzi spenti). N è la somma di 6 (blu) con 8(= modo multicolore). La variabile R è l'indirizzo di 'riferimento' dell'UFO nel codice dei caratteri della RAM. Qui è nella nona colonna della fila inferiore dello schermo. La linea 20 manda il microprocessore alla subroutine delle linee da 1000 a 1030, che introduce N nel codice dei colori della RAM, così che il colore di primo piano sarà blu, lo stesso del colore di fondo.

La linea 30 fissa il colore ausiliario sul giallo (codice = 7) introducendo 112 (= 7*16) ad inizializzare 36878. Quando l'UFO viene visualizzato, ha le luci accese. Il microprocessore passa poi alla subroutine della linea 2000. Questa subroutine ha uno scopo diverso da quello della precedente della linea 1000. Indirizza nelle loro corrette posizioni i tre caratteri, da X a

Z, nel codice RAM dei caratteri, utilizzando R come indirizzo del carattere X. Quando poi durante il programma verrà cambiato il valore di R, i caratteri verranno sempre visualizzati nelle loro rispettive posizioni. Possiamo così dire dove apparirà esattamente l'UFO sul video, semplicemente cambiando il valore di R. I valori dati dal calcolo 24+J sono i codici di schermo per ciascuno dei tre caratteri in questione.

L'effetto delle linee 20 e 30 è quindi quello di fare apparire l'UFO in fondo allo schermo, con i razzi spenti e le luci accese. Vi è un piccolo ritardo (linea 40) e poi le luci vengono spente facendo diventare il colore ausiliario da giallo a nero, il che significa, introdurre con un POKE il valore 0 nella cella 36878. A questo punto l'utente può interagire col programma battendo il tasto di funzione F7. Se questo non avviene, il programma salta alla linea 80, dove c'è un ulteriore ritardo, mantenendo le luci spente. La linea 90 riaccende le luci e si ritorna nuovamente alla linea 40 per ripetere la sequenza. Le luci si accendono e si spengono ripetutamente.

Se l'utente batte il tasto F7, entrano in azione i comandi nella linea 70. Assegnando ad N il valore 6, si rende blu il colore di primo piano, lo stesso dello sfondo. La subroutine alla linea 1000 introduce questo nelle celle dell'UFO, nel codice di colore della RAM. Notate che questa volta stiamo usando il codice dei colori per il blu (6), senza aggiungergli 8. L'effetto ottenuto è quello di disinnescare il modo multicolore per queste celle. Il risultato è che i caratteri vengono trattati come normali caratteri a due colori. Essi vengono visualizzati col blu in primo piano sul blu come colore di fondo (divengono invisibili!). Poi R viene diminuito di 22, per portare l'UFO una fila più in alto e la subroutine alla linea 2000 colloca i caratteri in questa fila. N viene reso uguale a 10 (2 per il colore rosso + 8 per il modo multicolore) e la macchina viene rimandata alla linea 20 per visualizzare l'UFO nella sua nuova posizione. Ora, col rosso come colore di primo piano, i razzi sono accesi. Ogni volta che battete F7, l'UFO risale di una fila e si aggira coi razzi incandescenti e le luci che lampeggiano minacciose contro il cielo notturno.

(Nota per coloro che non sanno resistere alla tentazione di aggiungere effetti sonori: Gli unici valori possibili dei toni che possono accompagnare il volo dell'UFO sono 214, 218, 162 e 193, realizzati in questo ordine dal generatore dei bassi. Se volete migliorare l'effetto, cambiate i colori delle luci dell'UFO in sincronia con ciascuna nota del tono. I colori sono (nell'ordine):giallo, rosa, porpora, blu e rosso).

In questo esempio, il movimento sullo schermo dell'oggetto in questione viene realizzato cambiando il valore del suo punto di riferimento R. Siamo anche in grado di cambiare contemporaneamente il colore di primo piano ed il colore ausiliario. Il solo cambiamento del colore ausiliario,

ottenuto con una singola istruzione di POKE, vi permette di cambiare l'aspetto a più oggetti collocati in punti differenti del video.

Il bambolotto danzante

Questo è un esempio di come usare il colore di primo piano e l'ausiliario per ottenere l'effetto di vedere parti mobili in un oggetto. Il bambolotto

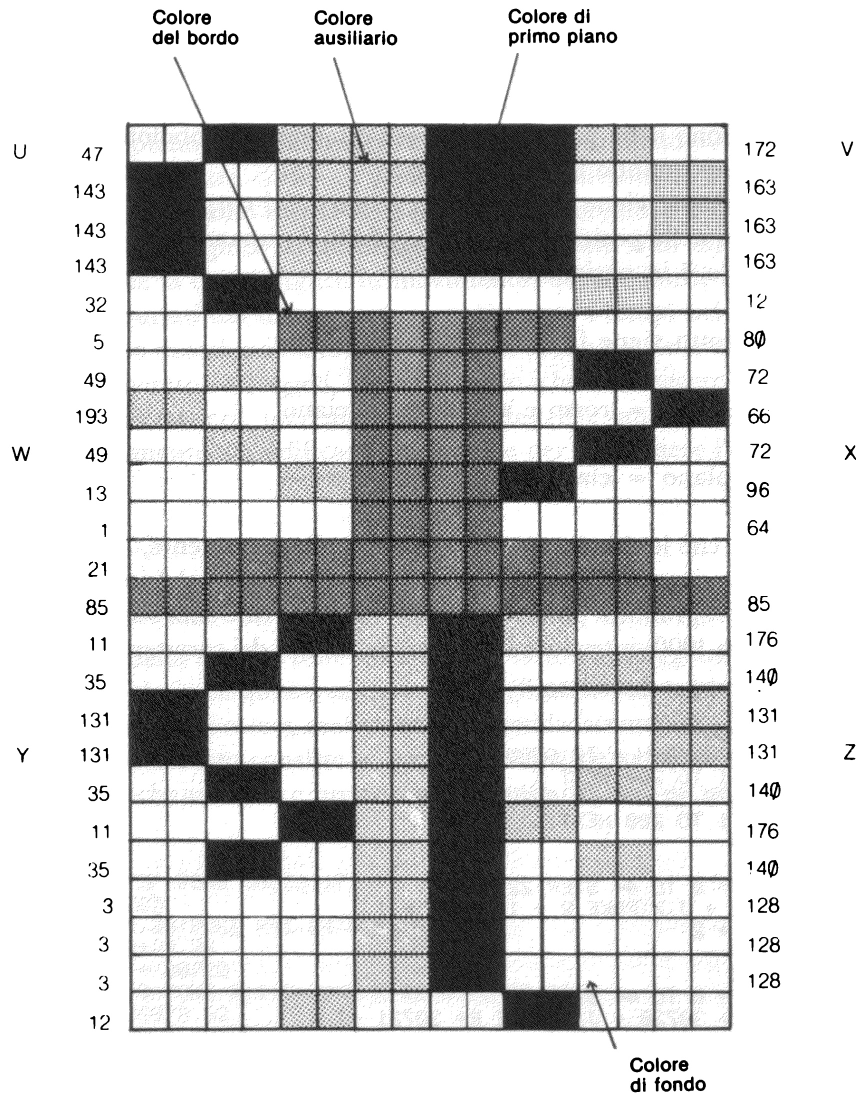


Fig. 11.5 - I caratteri utilizzati per il programma del pupazzo.

(fig 11.5) è rappresentato alternativamente in due posizioni (per descriverlo useremo l'espressione schermo sinistro e schermo destro):
(1) Testa rivolta a destra, braccio destro basso, braccio sinistro alto, gamba sinistra alzata.
(2) Testa rivolta verso sinistra, braccio sinistro basso, braccio destro alto, gamba destra alzata.

I quattro colori vengono così utilizzati:

Colore di fondo: come il fondo dello schermo (ciano).

Colore del bordo: viene usato per le parti del pupazzo che non si muovono, come il corpo e la gonna (rosso).

Colore di primo piano: viene utilizzato per le parti del corpo che si muovono quando sono nella posizione 1 (rosso quando il bambolotto è in posizione 1, ciano quando è in posizione 2).

Colore ausiliario: viene utilizzato per le parti del corpo che si muovono quando sono nella posizione 2 (rosso quando il pupazzo è in posizione 2, ciano quando è in posizione 1).

Il bambolotto viene fatto ballare passando da:

primo piano = rosso e ausiliario = ciano

a

primo piano = ciano e ausiliario = rosso

Questo fa sì che le due immagini appaiano alternatamente, dando l'illusione del movimento. Il programma viene mostrato nel Listato 11.14.

Come il programma precedente, esso contiene due subroutine. La prima (alla linea 1000) introduce i caratteri nel codice dei caratteri della RAM

```
10 PRINT"□":POKE 36879,58
20 R = 7800
30 GOSUB 1000
60 POKE 36878,48:N = 10:GOSUB 2000
70 FOR K = 1 TO 200:NEXT K
80 POKE 36878,32:N = 11:GOSUB 2000
90 FOR K = 1 TO 200:NEXT K
100 GOTO 60
1000 X = 21
1010 FOR J = 0 TO 44 STEP 22
1020 POKE R + J,X:POKE R + 1 + J,X + 1
1030 X = X + 2
1040 NEXT J
1050 RETURN
2000 FOR J = 0 TO 44 STEP 22
2010 POKE R+ 30720 + J,N:POKE R+ 30721 +J,N
2020 NEXT J
2030 RETURN
```

Listato 11.14 - Il pupazzo, un altro programma che sfrutta la grafica multicolore per ottenere l'animazione.

nella posizione determinata dall'indirizzo di riferimento R. L'altra (alla linea 2000) introduce il codice dei colori della RAM per controllare il colore di primo piano (N) e passare al modo con più colori.

La linea 20 inizializza l'indirizzo di riferimento e la linea 30 manda il microprocessore alla subroutine 1000 per introdurre i caratteri nella RAM. In questo caso una subroutine non sarebbe rigorosamente necessaria in quanto l'operazione viene fatta una sola volta. Ma se volete modificare il programma per fare muovere il pupazzo mentre balla, è utile poter cambiare R ed avere questa subroutine per fare riapparire il pupazzo nella sua nuova collocazione. In questo programma il bambolotto balla in un punto fisso ed una volta che i caratteri sono stati memorizzati nella RAM, tutto ciò che bisogna fare per dare inizio alla danza è scambiare i colori di primo piano ed ausiliario tra loro. Le linee da 60 ad 80 realizzano ciò, con dei ritardi tra ciascuna posizione dati dalle linee 70 e 90.

I ritardi vengono introdotti per dare all'osservatore il tempo di vedere ciascuna immagine. In un programma di giochi o di un qualsiasi altro genere che fa uso di figure in movimento, questi ritardi vengono pienamente sfruttati dal microprocessore. Durante i tempi nei quali l'oggetto è fermo, la macchina ha tutto il tempo di occuparsi delle altre routine del gioco: generare altri oggeti, muoverli per lo schermo, leggere eventuali istruzioni dalla tastiera, incrementare il punteggio e tante altre cose. Ritorna poi brevemente a modificare l'immagine per mantenere l'oggetto in movimento.

I mostri

Il limite della tecnica testè vista sta nel fatto che l'oggetto deve essere di un solo colore. In questo esempio finale (fig. 11.6) vi mostriamo come potete ottenere due colori, mantenendo un soddisfacente livello di movimento. Qui il movimento consiste in un 'c'è' e 'non c'è' o 'appare' e 'scompare' piuttosto che un vero e proprio movimento di parti da una posizione ad

```
10 PRINT"J":POKE 36879,172
20 R = 7705
30 POKE R,23:POKE R+1,24:POKE R+22,25
35 POKE R+23,26
40 R = R + 30720
50 POKER,8:POKE R+1,8:POKE R+22,8:POKER+23,8
70 POKE 36878,32
80 FOR K = 1 TO 200:NEXT K
90 POKE 36878,160
100 FOR K = 1 TO 200:NEXT K:GOTO 70
```

Listato 11.15 - Inizializzate il mostro ed esso sembra muoversi.

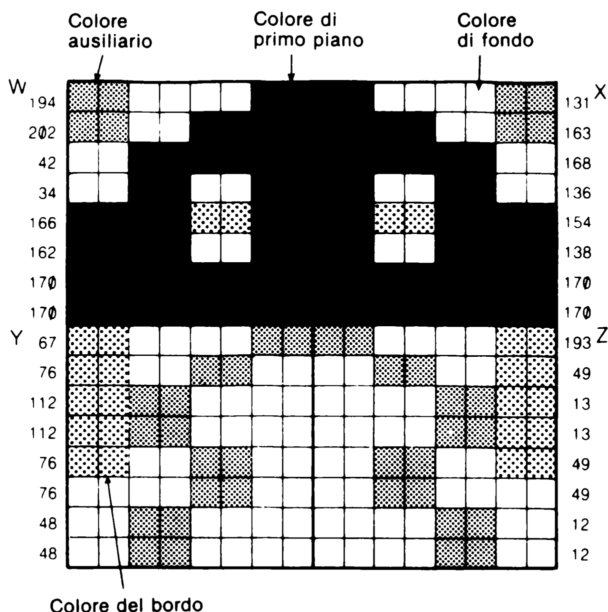


Fig.11.6 - I caratteri usati per rappresentare uno dei mostri.

un'altra. Il mostro ha sotto due tentacoli striscianti e sopra due piccole antenne, che protendono e si ritraggono alternatamente. Questi sono in colore ausiliario. Alternando il loro colore prima rosso e poi rosa (lo stesso dello sfondo), i tentacoli e le antenne vengono fatte apparire e scomparire.

Animare i disegni è solo una questione di introdurre due diversi valori alternatamente all'indirizzo 36878 (vedere il Listato 11.15). I caratteri vengono introdotti una volta per tutte nelle linee da 30 a 50. Questo riduce l'animazione nel modo più semplice e veloce, realizzando tutto ciò nelle linee 70 e 90. Le linee 80 e 100 generano i dovuti ritardi. Essendo così poco impegnato sul fronte dell'animazione, il microprocessore ha molto tempo da dedicare alle altre funzioni connesse al programma principale del quale i mostri sono solo una parte.

La prima parte del programma introduce i caratteri nel codice dei caratteri della RAM e nel codice dei colori della RAM una volta per tutte. Il corpo del mostro è in colore di primo piano (nero, così introduciamo 0+8=8 nella linea 50). I suoi occhi ed i due tentacoli più corti sono nel colore del bordo, porpora. Potete fare cambiare colore al corpo del mostro reinserendo la linea 50 con un altro valore nel POKE.

```
20 FOR R = 7705 TO 8025 STEP 80
```

Listato 11.16 - Queste due linee mostrano come ottenere un'orda di mostri!

```
60 NEXT R
```

Listato 11.17 - Per ottenere mostri animati

Per finire in bellezza, perchè non avere un'orda di mostri sparsa per lo schermo? Cambiate la linea 20 come indicato nel Listato 11.16, ed aggiungete la linea 60 (come nel Listato 11.17).

Ora avete sullo schermo una mezza dozzina di mostri, animati!

Riassunto

In questo capitolo avete scoperto come:

— animare un'immagine visualizzando due disegni diversi alternatamente
— animare un'immagine e farla muovere allo stesso tempo per lo schermo
— esaminare la possibilità di eventuali collisioni
— spostare un oggetto per lo schermo cambiandone il colore
— animare immagini usando il modo multicolore.

Capitolo 12
L'organizzazione del programmatore

Questo capitolo ricopre una gamma di argomenti che ci auguriamo vi sarà utile per ottenere il massimo dal vostro VIC 20.

Le REM

Molti lettori si chiederanno certamente perchè abbiamo accennato a questa istruzione così tardi. Tante pagine di programmi senza neppure una REM! Per rimediare alla mancanza, un esempio del suo uso vi viene proposto nel Listato 12.1.

```
10 REM QUESTO PROGRAMMA E' STATO SCRITTO PER
20 REM IL MICROCOMPUTER VIC 20
30 LET X = 99
40 PRINT X
```

Listato 12.1 - L'uso delle REM.

In questo programma vi è molto di più da dire, di quanto voi vi possiate immaginare. Fissiamo prima l'attenzione alle REM. Il microprocessore non le considera mai. REM è l'abbreviazione di 'REMark' (= commento) ed il commento è indirizzato a chi legge il programma,non al computer. Tutte le volte che la macchina trova la parola 'REM' in una linea di un programma, ignora il resto della linea e passa alla successiva. Nel programma qui sopra, il VIC non incomincia ad interessarsi al programma finchè non è arrivato alla linea 30.

Vi sono opinioni profondamente discordi sull'uso delle REM. Alcuni sono convinti assertori della loro utilità. Essi cominciano ogni loro programma con linee di REM che danno il titolo del programma, il suo scopo, spesso il nome dell'autore ed anche istruzioni dettagliate sull'uso. Pos-

sono poi avere altre REM sparse per il programma, che spiegano brevemente cosa succede ad ogni stadio di esso. Ogni subroutine comincerà con una REM che spieghi quello che fa. Non vi è alcun dubbio che le REM possono essere di aiuto ad altri per leggere e capire il programma. Le REM possono essere anche utili per ricordarvi come funzionano i vostri stessi programmi. Nonostante il fatto che questi vi possono essere molto chiari quando avete appena finito di scriverli, è sorprendente la rapidità con cui ci si dimentica molti dettagli. Anche solo una settimana dopo la stesura, il programma può sembrare incomprensibile perfino a voi che lo avete creato!

Un motivo per cui non vi sono REM nei programmi contenuti in questo libro sta nel fatto che il testo spiega i programmi molto meglio di quanto possa fare una qualsiasi REM. Altro motivo è che, siccome molti di essi sono esempi molto corti da introdurre e verificare all'istante, non vi è alcun motivo per darvi tutti quei caratteri aggiuntivi da battere. Un'altra argomentazione per limitare il più possibile l'uso delle REM, è che occupano molta memoria. Mentre infatti le istruzioni del BASIC vengono memorizzate in codice, con ciascuna parola che occupa un byte, ogni singola lettera, cifra, spazio o simbolo contenuto in una REM, occupa tutto un byte. Se avete a disposizione solo circa 3.5 kilobyte, non vi interessa sprecare buona parte di essi in un'informazione che il computer ignora regolarmente. Altro punto a sfavore delle REM è che, anche se tutta una linea del programma è REM ed il computer non se ne fa nulla, deve pur sempre leggere la linea fino a che non trova la parola REM. Questo richiede del tempo. Troppe REM rallentano la velocità alla quale gira il programma.

Da quanto detto finora risulta evidente che vi sono opinioni contrastanti sulle REM. Alcuni programmatori sono nettamente favorevoli, altri decisamente contrari. Noi vi abbiamo presentato entrambi i risvolti della medaglia, così che voi possiate farvi un'opinione personale sulla questione. La scelta più saggia si trova, forse, nella via di mezzo. Una REM all'inizio del programma, come titolo, è sempre utile per identificare il programma stesso. Vi aiuta anche a ricordarvi qual è il suo titolo esatto, dandovi così il nome corretto per caricarlo. Le REM sono anche utili per separare le sezioni principali del programma, soprattutto quando vi è un grosso salto da una parte all'altra del programma, o ad una subroutine. Ad ogni modo è buona regola non fare mai un salto ad una linea che comincia con REM, o non cominciare mai una subroutine con REM. È sempre meglio mettere le REM una linea prima della parte del programma cui si salta, o dell'inizio della REM (vedere il Listato 12.2).

Il motivo di ciò è che se state scrivendo il programma e volete omettere le REM per risparmiare spazio in memoria o fatica, è cosa semplice lasciare fuori una linea come 999. Se le REM fossero state sulla linea 1000

```
999 REM CODICI CARATTERE NELLA RAM
1000 FOR J = 0 TO 3
1010 READ X
```

Listato 12.2 - Dove mettere le REM all'interno dei vostri programmi.

e voi aveste tralasciato questa, non vi sarebbe più alcuna linea alla quale il microprocessore possa saltare quando venisse chiamata la subroutine. La cosa vi provocherebbe un messaggio di errore. Quando omettete una linea, dovete andare a riesaminare tutto il programma, correggendo tutte le istruzioni di GOTO e GOSUB relative. Questo evita possibili fonti di errori.

Nonostante voi siate concordi con la teoria favorevole alle REM, o con quella contraria, è essenziale che documentiate opportunamente ogni programma che scrivete. Se preferite tralasciare le REM, scrivete su carta tutte le vostre note. Annotatevi i passi principali del programma, cosa fanno e come funzionano. Se preferite, fatevi uno schema di flusso (vedere le figg. 5.6, 6.1 e 8.1) per evidenziare l'ordine con cui si succedono i passi principali. Talvolta si dice che ogni programmatore deve sempre fare uno schema di flusso prima di cominciare a scrivere un programma. Molta gente lo trova utile ed esistono certi programmi o parti di programma per i quali è quasi fondamentale incominciare col disegnare lo schema di flusso. In pratica però molti programmatori preferiscono pianificare i loro programmi scrivendo la successione dei passi principali con piccole frasi, magari aiutandosi con frecce che collegano le varie parti. Lo schema di flusso verrà disegnato dopo, una volta scritto e sperimentato il programma, come un bilancio finale di tutte le sue funzioni principali. 'Schemi di flusso prima' o 'schemi di flusso dopo' è un'altra scelta che dovete fare. Pensateci, scoprite quale dei due sistemi vi è più congeniale ed adottatelo.

Quando prendete nota del vostro programma, scorretelo dall'inizio alla fine, segnandovi il nome di ogni variabile e la sua funzione. È utile disegnare anche un diagramma di ogni matrice (vedere fig. 9.2) per mostrare cosa si suppone che contenga ciascuna riga e ciascuna colonna.

LET X = 99

Questa breve linea presa dal programma del Listato 12.1, è il prossimo argomento di discussione. LET è un'altra istruzione BASIC la cui presentazione è stata rimandata fino a questo capitolo. Il motivo è che non è indispensabile usarla. Se voi la scrivete, il VIC la comprende e la realizza. Se voi la lasciate fuori e scrivete semplicemente:

X = 99

il VIC si comporta come se ci fosse effettivamente LET e 'lascia' che X diventi uguale a 99. La maggior parte dei microprocessori, ma non tutti, non vi obbligano ad usare la parola LET in tali espressioni. È forse un peccato in quanto si tratta di una parola veramente importante. Osservata questa linea:

```
10 N = N + 3
```

Abbiamo usato spesso in questo libro linee di questo tipo. Diciamo che essa incrementa N di 3. Ma, guardandola meglio, si tratta di un controsenso! Come può N essere uguale ad N+3? Le regole della matematica ci insegnano che una tale equazione è impossibile da risolvere. Se però battessimo l'intera istruzione, come la interpreta il VIC, il vero significato diverrebbe più chiaro:

```
10 LET N = N + 3
```

Il significato quindi è: 'lascia che la variabile N assuma il valore che si ottiene incrementando di tre il valore attuale'. L''uguale' contenuto nell'espressione non significa 'uguale' nel senso che noi generalmente intendiamo in matematica. Esso viene utilizzato per assegnare un nuovo valore ad N. Per ottenere correttamente lo stesso significato, dovremmo sostituire l'uguale con il verbo 'divenire': 'lascia che N diventi la somma di N con 3'.

Potremmo facilmente liberarci di questa confusione sul segno di uguale se questo fosse usato solo col significato che assume in BASIC, ma non è così. In un'espressione del tipo: 'IF X=5 THEN...' esso ha il suo abituale significato matematico, confrontando una grandezza con un'altra-viene infatti usato come operatore relazionale. È importante che voi abbiate ben chiaro in testa questi due usi distinti del segno di uguale.

Gran parte dei programmatori tralasciano LET, quando i loro microprocessori permettono loro questo. Vi sono anche altre parole che in certi casi possono essere tralasciate. Il non scriverle rende il programma un po' meno leggibile, soprattutto per un principiante, ma chi è solo appena abituato già lo fa. Per esempio, non è generalmente necessario usare la parola 'GOTO' in una istruzione del tipo 'IF...THEN GOTO numero della linea'. Al posto di:

```
20 IF X = 5 THEN GOTO 300
```

è corretto scrivere:

```
20 IF X = 5 THEN 300
```

Il microprocessore interpreta i numeri che seguono THEN come l'indirizzo di una linea.

```
10 FOR J = 1 TO 5
20 FOR K = 3 TO 6
30 FOR L = 10 TO 3 STEP -2
40 PRINT J*K*L
50 NEXT
60 NEXT
70 NEXT
```

Listato 12.3 - Potete omettere i nomi delle variabili dopo NEXT.

Altra omissione compatibile col computer è la variabile che segue un'istruzione di NEXT. Guardate per esempio il Listato 12.3. Il microprocessore non si preoccupa della mancanza dei nomi delle variabili nelle linee da 50 a 70. Questo programma può inoltre essere reso più compatto scrivendo tutti i NEXT in un'unica linea come nel Listato 12.4.

```
50 NEXT:NEXT:NEXT
```

Listato 12.4 - Tutte le istruzioni NEXT possono essere collocate in un'unica linea del programma.

Nonostante sia semplice per il lettore seguire il percorso dei NEXT in un programma breve e semplice come il precedente, la cosa diventa molto più problematica in un programma più lungo e con molti loop. Il computer tiene automaticamente conto di quanti FOR e quanti NEXT avete scritto nel programma e vi informa, con un messaggio di errore, se ne avete messi troppi degli uni o degli altri. Una persona che controlla il programma può invece facilmente confondersi. Così, a meno che non vogliate risparmiare fino all'ultimo byte di memoria, è preferibile mantenere i nomi delle variabili dopo ogni next.

Risparmiare spazio in memoria

Nonostante i nuovi microprocessori possiedano delle memorie di dimensioni che solo alcuni anni fa sarebbero state incredibilmente costose, vi è una variante alla legge di Parkinson che dice che un programma si espande occupando tutto lo spazio RAM disponibile per contenerlo. Solo alcune routine aggiuntive ed altri raffinamenti ad un programma già 'perfetto' fanno sparire altri due kilobyte! È così buona cosa osservare alcune norme per risparmiare spazio in memoria.

(1) Omettete tutte le REM non strettamente essenziali.
(2) Omettete gli spazi tra le parole nelle linee del programma. Nei listati contenuti in questo libro, sono stati usati liberamente gli spazi per rendervi più semplice la lettura dei programmi, ma una linea del tipo:

```
200POKE36879,27:B(8,N)=5632+8*ASC(K$(N)):N=N+1
```

viene facilmente compresa dal VIC, anche se a voi può sembrare indigesta. Tralasciare gli spazi vi permette di scrivere un maggior numero di istruzioni in una stessa linea (vedere il punto (3)).
(3) Usate le linee contenenti più istruzioni (per esempio, quella appena vista nel Listato 12.4). Le istruzioni vengono separate l'una dall'altra con i due punti (:). Il VIC vi permette di scrivere fino ad 84 caratteri in una linea, compresi gli spazi, la punteggiatura ed i simboli, ma non compreso lo spazio che appare automaticamente (sul listato) tra l'etichetta della linea ed il primo carattere scritto su di essa.
(4) Usate le linee e le colonne 'zero' delle matrici. La riga in testa alla matrice A() in fig. 9.2 è uno spreco di RAM, in quanto gli zeri vengono caricati in memoria anche se non vengono mai usati nel programma. Sarebbe stato più economico usare anche questo spazio, anche se avrebbe reso l'elaborazione del loop che lavora su questa matrice un po' più difficile da capire. Quando sarete abituati a lavorare con le matrici, vi sarà più facile usare anche le righe e colonne 'zero' senza fare confusione. L'uso delle righe e delle colonne 'zero' è particolarmente importante se lavorate con matrici molto grosse.
(5) Usate le subroutine invece di ripetere la stessa successione di istruzioni più volte in un programma.
(6) Usate le variabili intere (Capitolo 4). Queste richiedono un spazio di memoria molto minore delle variabili razionali, che fanno uso della virgola mobile. Potete ottenere grossi risparmi se nei vostri programmi avete grandi matrici. Per esempio, se estendete la funzione ausiliaria FXG (Listato 9.1) a comprendere molte più fasi, è molto meglio definire la matrice A() come matrice A%(), in quanto tutti i numeri necessari per controllare i generatori di suono sono interi.

Velocizzare i programmi in BASIC

Vi sono alcuni modi per fare girare un programma il più veloce possibile.

(1) Tralasciare tutte le REM che non sono strettamente indispensabili. Questo non è tanto importante nelle prime linee di un programma, che ven-

gono probabilmente lette una sola volta dal microprocessore. Le REM che si trovano invece nelle linee che vengono lette molte volte (ad esempio in un loop tipo FOR...NEXT) possono rallentare considerevolmente l'esecuzione di un programma. Lo stesso dicasi per le REM che si trovano all'inizio di una subroutine che viene spesso chiamata (altro motivo per metterle nella linea precedente).
(2) Definite per prime nel programma le variabili più comunemente usate, poi quelle più raramente interessate. Il motivo di ciò è il fatto che il microprocessore memorizza in RAM i valori di ciascuna variabile. Memorizza le variabili nell'ordine in cui compaiono per la prima volta nel programma. Ogni volta che deve cercare il valore di una variabile, o cambiarlo, scandisce una lista per trovare la variabile. Le variabili che si trovano in alto nella lista vengono trovate più velocemente delle altre più in basso nella lista. Questo è particolarmente importante per quelle variabili che vengono utilizzate nei loop e che possono venire ricalcolate migliaia di volte durante l'esecuzione di un programma. È semplice dichiarare queste variabili all'inizio di un programma, anche se non vengono effettivamente utilizzate a quel punto del programma, inserendo una linea del tipo di:

```
10 N = 0:X = 0:P = 0
```

In questa linea, N dovrebbe essere la variabile più usata di tutte. Dobbiamo interessarci a molte di queste tecniche per risparmiare tempo, quando vogliamo velocizzare al massimo programmi molto lunghi, che fanno uso di molte variabili.

```
10 FOR J = 1 TO 100
20 PRINT 1234 * 5678
30 NEXT J
```

Listato 12.5 - Un loop nel quale il VIC deve convertire gli operandi, prima di moltiplicarli.

(3) Usate nei loop le variabili e non i valori numerici. Per vedere la differenza che c'è, consideriamo l'esempio nel Listato 12.5. Ci mette 3.5 secondi per girare. Ad ogni passaggio nel loop il VIC deve convertire le cifre '1234' e '5678' nei rispettivi valori in numeri binari prima di poter fare la moltiplicazione e stampare il risultato. Il programma del Listato 12.6, che dà lo stesso risultato, impiega solo 2.82 secondi per girare, una riduzione del 18%.

```
5 X = 1234: Y = 5678
10 FOR J = 1 TO 100
20 PRINT X*Y
30 NEXT J
```

Listato 12.6 - Qui la conversione viene fatta prima dell'inizio del loop.

I due numeri vengono calcolati in binario alla linea 5 una volta per tutte. Alla linea 20, il VIC deve solo leggere i valori di X e di Y nella sua tabella delle variabili prima di elaborare il risultato. L'osservare i valori nella tabella delle variabili è dunque più veloce che convertire i numeri ogni volta nel loop.

Interagire con la macchina

L'utente può interagire col computer in due maniere: quando scrive qualcosa sulla tastiera e quando legge qualccosa sul video. I vostri programmi dovrebbero essere fatti in modo che sia facile per l'utente comunicare attraverso la tastiera col computer e per il computer comunicare attraverso il video con l'utente.

Spesso chi vuole usare con piacere il computer, non è esperto di calcolatori e tantomeno è esperto in dattilografia, così diventa essenziale rendere l'uso della tastiera il più semplice e meno faticoso possibile. In primo luogo bisogna dire all'utente molto chiaramente che cosa deve esattamente fare ad ogni passo. Le istruzioni possono avere la forma di un manuale, qualcosa di scritto o di stampato, o possono essere visualizzate sullo schermo. Con il VIC senza espansione di memoria non vi è generalmente lo spazio per avere più videate di istruzione, ma anche pochi byte di memoria possono essere ben impiegati. Esempi di utili messaggi di INPUT sono:

Volete continuare il gioco (Y/N)?
Quanti giocatori (1-4)?
Che mese (1-12)?
Battete qualsiasi tasto per continuare

Il primo esempio indica all'utente che deve rispondere con un sì o con un no e che la risposta viene correttamente data battendo o il tasto 'Y' o il tasto 'N'. Non ha senso richiedere all'utente di battere le intere parole 'Yes' o 'No', quando le sole iniziali sono perfettamente chiare. Il secondo esempio dice all'utente qual è il numero massimo di giocatori ammesso. Il terzo specifica che il mese in risposta non deve essere battuto come una parola, ma col numero d'ordine. Si chiederebbe troppo da un non dattilografo di battere 'febbario', scusate, 'febbraio', quando è sufficiente il numero '2'.

Il VIC ha la possibilità di ricevere dati dall'utente sia con INPUT, che con GET (o GET$). GET e GET$ possono essere usati solo quando deve essere battuto un solo tasto. Potreste usare GET$ per ricevere 'Y' o 'N', qualsiasi numero nell'intervallo tra 0 e 9, o 'qualsiasi tasto' nel quarto de-

gli esempi qui sopra. GET$ viene anche usato nei giochi per sparare, muovere mazze e molte altre operazioni, spesso in unione con i tasti di funzione.

È meglio non utilizzare GET (o GET$) ed INPUT nello stesso programma. INPUT richiede che l'utente batta anche il tasto di RETURN, mentre GET e GET$ no. Questo potrebbe confondere l'utente. Se doveste utilizzarli entrambi nello stesso programma, visualizzate l'istruzione di battere RETURN ogni volta che viene usato INPUT. Allo stesso modo, se un'istruzione di INPUT richiede alcuni numeri come risposta, spiegate all'utente come bisogna inserire più numeri e ricordategli di separarli con le virgole. La stessa risposta del microprocessore ad errori in ingresso, '? REDO FROM START' non è incoraggiante.

Controllare ciò che viene introdotto

Sebbene aiutato, l'utente potrebbe introdurre dei caratteri sbagliati. Generalmente questo è involontario, ma vi è anche chi si diverte ad introdurre espressioni ridicole per vedere se il computer sa trattare anche queste. Il controllo degli input vi aiuta ad evitare che il programma si blocchi accidentalmente o volontariamente. Se deve essere introdotto un numero compreso in una certa gamma, è facile controllarlo (vedere il Listato 12.7).

```
200 PRINT"QUANTI GIOCATORI? (1-4)":INPUT N
210 IF N<1 OR N>4 THENGOTO 215
215 PRINT"IMMETTI UN NUMERO TRA 1 E 4":GOTO200
```

Listato 12.7 - Controllate i valori dati in INPUT, per assicurarvi che siano compresi nel dovuto intervallo.

Notate la gentilezza della domanda- un microprocessore ha una pazienza infinita e può dunque essere gentile. Notate inoltre come il messaggio della linea 210 dia delle spiegazioni più esplicite su come rispondere correttamente. Altro punto da rilevare è il messaggio che viene visualizzato tramite un'istruzione di PRINT. Messaggi particolarmente corti possono essere inseriti come parte di un INPUT, ma il VIC non può accettare dei messaggi più lunghi di 20 caratteri in un'istruzione di INPUT. Messaggi più lunghi possono provocare un errore, o parte di essi può misteriosamente comparire nella stringa di risposta.

Se obiettate che il vostro programma ha tanti INPUT e che per avere un particolare messaggio di errore per ognuno di essi comporterebbe troppo utilizzo di memoria, provate ad usare lo stesso messaggio per ogni INPUT. Mettete all'inizio del programma la linea:

10 M$='PER FAVORE, INTRODUCI UN NUMERO TRA'

La linea successiva all'INPUT della linea 200 potrebbe essere:

210 IF N<0 OR N>4 THEN PRINT M$;'1 E 4':GOTO 200

Potete così usare diversi estremi per l'intervallo assegnato di volta in volta a ciascun INPUT.

Simili controlli possono essere estesi anche ad ingressi di tipo stringa.

```
320 IF A$<>"N" THEN PRINT"RISPONDI 'S' O 'N'"
330 GOTO 300
```

Listato 12.8 - INPUT di una stringa di controllo.

Una richiesta tipo 'si/no' può essere verificata come mostrato nel Listato 12.8. La domanda viene scritta prima in una PRINT, in quanto è troppo lunga per elaborarla con un INPUT. In questo esempio, il gioco finisce a meno che l'utente non batta 'Y'. In nessun punto l'utente viene forzato a rispondere 'Y' o 'N'. Dopo tutto è poco male se l'utente batte 'T' al posto di 'Y', in quanto il gioco può facilmente venire riattivato facendo girare di nuovo il programma. Se invece fosse essenziale la conferma della risposta negativa, potreste cambiare la linea 320 del Listato 12.8 come indicato nel Listato 12.9.

```
400 PRINT"CANCELLI TUTTI I DATA?(S/N)":INPUTA$
410 IF A$="S"THEN INPUT"NE SEI SICURO?";A$
420 IF A$="S"THEN GOSUB 1000
```

Listato 12.9 - La richiesta di conferma in caso di risposta negativa.

Talvolta è importante sensibilizzare l'utente sull'importanza della risposta, che potrebbe avere conseguenze drastiche. Il secondo INPUT (linea 410) contiene un breve messaggio, perciò non deve essere stampato separatamente sul video. Se l'utente risponde due volte 'Y', questo può essere rassicurante ed il programma può procedere cancellando i dati. Altrimenti il programma passa alla linea 430 ed alle successive, mantenendo intatti i dati.

```
300 PRINT"VUOI GIOCARE ANCORA?":INPUT A$
310 IF A$ = "S" THEN 200
320 END
```

Listato 12.10 - Come dare all'utente la possibilità di cambiare idea.

Un problema che nasce con gli INPUT è che se come risposta ci si aspetta un numero ed invece viene data una lettera, il computer non la accetta.

```
500 INPUT"CHE MESE? (1-12)";A$
510 A = VAL(A$)
520 IF A<1 OR A>12 THEN GOTO 525
525 PRINT"INTRODUCI UN NUMERO TRA 1 E 12"
530 GOTO 500
```

Listato 12.11 - Uso delle stringhe in INPUT per controllare dei numeri.

Appare sul video il messaggio: 'REDO FROM START' (=riparti dall'inizio). È facile che un principiante batta al posto di un numero una delle lettere dei tasti della linea più in alto sulla tastiera, o che scriva 'TRE' quando il computer si aspetterebbe '3'. Un modo per prevenire questo tipo di errori è assumere tutti gli INPUT come delle stringhe (vedere il Listato 12.11). Questa routine è leggermente più lunga di quella data precedentemente. VAL(A$) assegna il valore zero a tutte le risposte che fanno uso dell'alfabeto. La routine chiede di reintrodurre la risposta nel caso siano state introdotte lettere o cifre al di fuori dell'intervallo assegnato.

L'unico problema che può nascere con i controlli è che le relative routine possono facilmente occupare più linee del programma della parte principale del programma stesso. Nei programmi a carattere matematico, nei quali è essenziale che i dati vengano rigorosamente controllati prima di lavorarci sopra, le routine relative agli ingressi arrivano ad occupare anche i tre quarti del programma. Il vero problema consiste nel trovare un giusto compromesso tra il controllare gli input, da una parte, e, dall'altra, mantenere il programma di dimensioni ragionevoli.

I programmi applicativi presentati in questo libro non hanno alcun tipo di controllo sugli ingressi, non prendeteli quindi come esempi da seguire! Il vantaggio che ne abbiamo tratto sta nel lavoro che vi abbiamo evitato di fare la prima volta che avete dovuto introdurli nella macchina. Ammesso che voi li abbiate ora registrati su di un nastro, questa può essere una buona occasione per sperimentare i consigli che vi abbiamo appena dato e aggiungere ad essi per conto vostro alcune routine di controllo degli INPUT.

Gli output

Il microprocessore è in grado di comunicare con voi sia tramite il video che tramite i suoni dei generatori. È facile programmare il VIC per generare un suono che attragga l'utente. Questo dovrebbe preferibilmente essere un tono, o una serie di toni. Per esempio quelli utilizzati negli aeroporti per gli annunci pubblici. Questo è sicuramente preferibile al perentorio 'bip' che il computer generalmente emette!

Anche i colori giocano il loro ruolo nel guidare l'utente. Nei programmi applicativi Newkeys e Multikeys (Listati 7.3 e 7.4), il colore del bordo

diviene verde per indicare che il nuovo carattere è stato accettato e vengono calcolati i codici relativi. Questo è certamente meglio di una inutile pausa durante la quale l'utente si chiede se tutto procede per il verso giusto, o se, per errore, ha battuto un tasto sbagliato. In altri programmi che prevedono tempi di calcolo abbastanza lunghi, è meglio inserire un messaggio del tipo: 'STO CALCOLANDO - PER FAVORE ATTENDI' da visualizzare nel frattempo. In questo modo l'utente è informato che qualcosa sta succedendo e non si preoccupa che il programma si sia fermato per qualche ignoto motivo.

Altro modo in cui il colore può essere utile è indicando a quale tappa del programma si è arrivati. Ad un passo lo schermo può essere rosa, col bordo blu e le lettere rosse. Al passo successivo, potrebbero essere utilizzati altri colori. La colorazione aiuta così l'utente a capire a che punto del programma è arrivato ed il tipo di risposte che la macchina si attende da lui. Il colore può anche attirare l'attenzione sulle intestazioni o su particolari messaggi visualizzati in un colore diverso dal resto del testo. Con un computer versatile come il VIC, non vi sono scuse per eventuali display smorti e poco interessanti.

Riassunto

In questo capitolo avete trovato come:

— utilizzare (o non utilizzare) le REM
— arricchire i vostri programmi di una documentazione per voi e per altri
— fare economia di spazio in memoria
— rendere i vostri programmi il più veloce possibile
— manipolare gli ingressi dalla tastiera
— rendervi ancora più amico il vostro VIC, di per sé già amichevole.

Appendice A
Tabelle utili

Tavola 1 Cosa fanno i tasti di controllo.
Questi tasti hanno la stessa funzione in entrambi i modi.

Tasto	*Cosa fa*
CLR/HOME	Manda a home il cursore; con SHIFT cancella tutto quello che c'è sullo schermo.
CRSR (con le frecce)	Muove il cursore in basso ed a destra, con SHIFT lo sposta in alto ed a sinistra.
CTRL	Cambia colori (vedere la tavola 2); fa anche girare più lentamente un programma.
'Flag' (simbolo del Commodore)	Dà i simboli di sinistra (vedere la tavola 2). Se usato con SHIFT cambia il modo. Può essere usato al posto di SHIFT per controllare il cursore.
INST/DEL	Cancella caratteri; usatelo con lo SHIFT per inserirne di nuovi.
RETURN	Viene usato quando termina una istruzione, una linea, o un programma.
RESTORE	Battetelo con RUN/STOP per cancellare lo schermo e ripristinare lo stato iniziale del VIC appena acceso.
RUN/STOP	Ferma il VIC quando sta facendo girare un programma (vedere pagina 16). Usato anche con RESTORE (vedere sopra). Se usato con SHIFT dà il comando per caricare dal nastro un programma.
SHIFT	Cambia l'effetto di molti tasti (vedere sopra e la tavola 2). Se usato con 'Flag' cambia il modo.
SHIFT LOCK	Battuto e fissato ha lo stesso effetto che tenere lo SHIFT pressato. Premere di nuovo per togliere il LOCK.

Tavola 2 Cosa fanno gli altri tasti (modo della grafica)

Tasto	*Se usato da solo*	*Ciò che fa* con lo SHIFT	con 'Flag'	con CTRL
A - Z	A - Z	Set di destra	Set di sinistra	—
1 - 8	1 - 8	Simboli in alto*	Simboli in alto*	Cambia colore
9	9	)	)	Inserisce il modo inverso
0(zero)	0	0	0	Disinserisce il modo inverso
+ — £ @ *	+ — £ @ *	Set di destra	Set di sinistra	—
: ; , . /	: ; , . /	[ ] < > ?	[ ] < > ?	—
↑	↑	π	π	—
= ←	= ←	= ←	= ←	—
Barra spaziatrice	spazio	spazio	spazio	—

* I simboli segnati sui tasti al di sopra dei numeri.
I 4 tasti di funzione marroncini, alla destra della tastiera principale, vengono illustrati nel capitolo dieci

Tavola 3 Cosa fanno gli altri tasti (modo del testo)

Tasto	*Se usato da solo*	*Ciò che fa* con lo SHIFT	con 'Flag'
Da A a Z	Da a a z	Da A a Z	Set di sinistra
+ —	+ —	Set di destra	Set di sinistra
£	£	▨	Set di sinistra
@	@	✓	Set di sinistra
*	*	Set di destra	▧
↑	↑	▦	▦

I tasti qui non elencati svolgono la stessa funzione che espletano nel mondo della grafica (vedere la Tavola 2). I tasti di questa tavola non hanno alcun effetto se battuti con il CTRL.

Tavola 4 Ottenere ciò che desiderate

Effetto desiderato	*Cosa fare*
Mandare il cursore in 'home'	Battere CLR/HOME
Mandare il cursore in 'home' e cancellare lo schermo	Battere CLR/HOME con SHIFT
Muovere il cursore verso il basso o a destra	Usare i tasti CRSR
Muovere il cursore verso l'alto o a sinistra	Usare i tasti CRSR con SHIFT
Cambiare colore	Usare i tasti dei numeri da 1 a 8, con CTRL
Attivare il modo reverse	Battere il tasto 9 con CTRL
Disattivare il modo reverse	Battere il tasto 0 (zero) con CTRL
Cancellare un carattere	Battere INST/DEL (tenere premuto per cancellare più caratteri)
Cambiare modo	Battere SHIFT con 'Bandiera' (tasto Commodore)
Scrivere maiuscolo	Se in modo grafico, battere i tasti senza SHIFT, se in modo testo con SHIFT
Scrivere minuscolo	Usare l'alfabeto con il modo testo senza SHIFT
Usare i caratteri superiori	Battere il tasto con SHIFT
Usare i caratteri di sinistra	Battere il tasto con 'Flag' (*dipende dal modo, vedi Tavola 3)
Usare i caratteri di destra sui tasti dell'alfabeto	In modo grafico, battere il tasto con SHIFT
Usare i caratteri di destra sui tasti + — £ * @	Battere il tasto con SHIFT (per @ e £ dipende dal modo, vedi Tavola 3)
Reinizializzare il VIC	Battere RUN/STOP con RESTORE

Tavola 5 Impostazione del colore dello schermo e del bordo.

I valori contenuti in questa tabella sono quelli da introdurre con una POKE nell'indirizzo 36879 per fissare il colore dello schermo e del bordo. I codici di colore di ciascuno vengono dati tra parentesi dopo il nome di ogni colore.

				Colori del bordo				
Colori del video	*Nero* (0)	*Bianco* (1)	*Rosso* (2)	*Ciano* (3)	*Porpora* (4)	*Verde* (5)	*Blu* (6)	*Giallo* (7)
Nero (0)	8	9	10	11	12	13	14	15
Bianco (1)	24	25	26	27	28	29	30	31
Rosso (2)	40	41	42	43	44	45	46	47
Ciano (3)	56	57	58	59	60	61	62	63
Porpora (4)	72	73	74	75	76	77	78	79
Verde (5)	88	89	90	91	92	93	94	95
Blu (6)	104	105	106	107	108	109	110	111
Giallo (7)	120	121	122	123	124	125	126	127
Arancio (8)	136	137	138	139	140	141	142	143
Arancio chiaro (9)	152	153	154	155	156	157	158	159
Rosa (10)	168	169	170	171	172	173	174	175
Ciano chiaro (11)	184	185	186	187	188	189	190	191
Porpora chiaro (12)	200	201	202	203	204	205	206	207
Verde chiaro (13)	216	217	218	219	220	221	222	223
Blu chiaro (14)	232	233	234	235	236	237	238	239
Giallo Chiaro (15)	248	249	240	241	242	243	244	245

Nota: ciano è un colore verde-bluastro, talvolta detto 'turchese'.

Tavola 6 I codici di schermo delle lettere e dei simboli

Per alcuni codici, il carattere stampato dipende dal fatto che il VIC è in Modo Grafico o in Modo Testo. I simboli che compaiono tra le colonne del Modo Grafico e del Modo Testo sono uguali per entrambi i modi.

Codice	*Modo grafico*	*Modo testo*	*Codice*	*Modo grafico*	*Modo testo*
0	@		24	X	x
1	A	a	25	Y	y
2	B	b	26	Z	z
3	C	c	27	[	
4	D	d	28	£	£
5	E	e	29	]	
6	F	f	30	↑	
7	G	g	31	←	
8	H	h	32	(spazio)	
9	I	i	33	!	
10	J	j	34	"	
11	K	k	35	#	
12	L	l	36	$	
13	M	m	37	%	
14	N	n	38	&	
15	O	o	39	'	
16	P	p	40	(	
17	Q	q	41	)	
18	R	r	42	*	
19	S	s	43	+	
20	T	t	44	,	
21	U	u	45	–	
22	V	v	46	.	
23	W	w	47	/	

Tavola 6 Segue

Codice	*Modo grafico*	*Modo testo*	*Codice*	*Modo grafico*	*Modo testo*
48	Ø		76		L
49	1		77		M
50	2		78		N
51	3		79		O
52	4		80		P
53	5		81		Q
54	6		82		R
55	7		83		S
56	8		84		T
57	9		85		U
58	:		86		V
59	;		87		W
60	<		88		X
61	=		89		Y
62	>		90		Z
63	?		91		
64			92		
65		A	93		
66		B	94	π	
67		C	95		
68		D	96	(spazio)	
69		E	97		
70		F	98		
71		G	99		
72		H	100		
73		I	101		
74		J	102		

Tavola 6 Segue

Codice	*Modo grafico*	*Modo testo*	*Codice*	*Modo grafico*	*Modo testo*
75		K	103		
104			116		
105			117		
106			118		
107			119		
108			120		
109			121		
110			122		✓
111			123		
112			124		
113			125		
114			126		
115			127		

Tavola 7 I codici di colore per i caratteri visualizzati.

Colore	*Codice*
Nero	0
Bianco	1
Rosso	2
Ciano (turchese)	3
Porpora	4
Verde	5
Blu	6
Giallo	7

Tavola 8 I generatori di suono del VIC 20

Questa tavola mostra come calcolare la frequenza dei generatori di note (vedere anche la Tavola 9). In pratica, 128 genera la frequenza più bassa mentre 254 quella più alta.

Nome	*Indirizzo*	*Gamma di valori*	*Effetto dovuto all'indirizzare il numero N*
Note basse	36874	da 128 a 254	Frequenza=8192/(255—N)
Note medie	36875	da 128 a 254	Frequenza=16384/(255—N)
Note acute	36876	da 128 a 254	Frequenza=32768/(255—N)
Rumore	36877	da 128 a 254	Basso brontolio se N=128 Sibili se N=254
Volume	36878	da 0 a 15	Silenzio quando N=0 Massimo vol. quando N=15

Tavola 9 Note della scala musicale

Questi sono i valori da introdurre nei generatori di note. I toni non citati sono i semitoni compresi tra le note riportate nelle linee sopra e sotto.

Nota	Soprano	Tenore	Basso
F"	232		
E"	230		
	229		
D"	227		
	226		
C"	224		
B'	222		
	220		
A'	218		
	216		
G'	214		
	211		
F'	209	232	
E'	206	230	
	203	229	
D'	200	227	
	196	226	
C'	193	224	
B	189	222	
	185	220	
A	181	218	
	177	216	
G	172	214	
	167	211	
F	162	209	232
E	157	206	230
	151	203	229
D	147	200	227
	138	196	226
Medio C	131	193	224
B		189	222
		185	220
A		181	218
		177	216
G		172	214
		167	211
F		162	209

Tavola 9 Segue

Nota	Soprano	Tenore	Basso
E		157	206
		151	203
D́		147	200
		138	196
C		131	193
B			189
			185
A			181
			177
G			172
			167
F			162
E			157
			151
D			147
			138
C			131

Tavola 10 I codici ASCII

Questa tabella elenca i codici che vengono implementati in modo standard sul VIC.

Codice	*Carattere*	*Codice*	*Carattere*
13	(Return)	61	=
32	(Spazio)	62	>
33	!	63	?
34	"	64	@
35	#	65	A
36	$	66	B
37	%	67	C
38	&	68	D
39	'	69	E
40	(	70	F
41	)	71	G
42	*	72	H
43	+	73	I
44	,	74	J
45	-	75	K
46	.	76	L
47	/	77	M
48	0	78	N
49	1	79	O
50	2	80	P
51	3	81	Q
52	4	82	R
53	5	83	S
54	6	84	T
55	7	85	U
56	8	86	V
57	9	87	W
58	:	88	X
59	;	89	Y
60	<	90	Z

Tavola 11 I tasti funzionali ed i loro codici CHR$

Senza shift/con shift	*Tasto*	*Numero*	*Codice CHR$*
Senza shift	alto	F1	133
	secondo più basso	F3	134
	terzo più basso	F5	135
	basso	F7	136
Con shift	alto	F2	137
	secondo più basso	F4	138
	terzo più basso	F6	139
	basso	F8	140

Tavola 12 Programmi che utilizzano i simboli grafici definiti dall'utente (Metodo 2)

Numero dei caratteri	*Definisce come tasti*	*La linea 5 legge* 'FOR J = 0 TO...'	*La linea 7 legge* 'POKE...+J,X'
1	Z	7	6352
2	YZ	15	6344
3	XYZ	23	6336
4	WXYZ	31	6328
5	VWXYZ	39	6320
6	UVWXYZ	47	6312
N	ultimi N tasti	8*N — 1	6360 — 8*N

Tavola 13 Programmi che utilizzano i simboli grafici definiti dall'utente (Metodo 3)

Numero dei caratteri	*Definisce come tasti*	*La linea 5 legge* 'FOR J = 0 TO...'	*La linea 7 legge* 'POKE...+J,X'
1	Z	7	7376
2	YZ	15	7368
3	XYZ	23	7360
4	WXYZ	31	7352
5	VWXYZ	39	7344
6	UVWXYZ	47	7336
N	ultimi N tasti	8*N — 1	7384 — 8*N

Appendice B
Programmi con espansione di RAM

I programmi di questo libro sono stati scritti per i VIC 20 senza espansione di memoria, quando si hanno circa 3.5 kilobyte di memoria RAM a disposizione dell'utente. Se disponete di una ulteriore memoria RAM può essere necessario cambiare uno o due valori in alcuni programmi.

Se avete espanso la memoria RAM di 3K (3 kilobyte) e non oltre, non è necessario modificare i programmi. Non avete bisogno di leggere oltre.

Se invece avete espanso la memoria RAM di oltre 3K, il VIC colloca in nuovi indirizzi i suoi codici dei caratteri ed i suoi codici dei colori della RAM. Ciò significa che quei programmi grafici nei quali si introducevano dei codici di caratteri o di colori in certi indirizzi devono ora essere modificati per tenere conto dei nuovi indirizzi. I programmi che non indirizzano queste aree della RAM (e questo significa la maggior parte dei programmi contenuti in questo libro) non devono essere corretti.

I cambiamenti da fare agli indirizzi sono:

Il codice dei caratteri della RAM comincia ora dall'indirizzo 4096 (invece di 7680).

Il codice dei colori della RAM comincia ora all'indirizzo 37888 (invece di 38400).

Vi sono 3 tipi di cambiamenti da effettuare:

(1) Introducendo dei caratteri nel codice dei caratteri della RAM, diminuire di 3584 tutti gli indirizzi, così che partano ora da 4096.
(2) Introducendo dei colori nel codice dei colori della RAM, diminuire di 512 tutti gli indirizzi, così che partano ora da 37888.
(3) Alcuni programmi fanno uso del valore 30720 per calcolare gli indirizzi del codice dei colori della RAM dai corrispondenti indirizzi nel codice dei caratteri della RAM; in questi programmi sostituite a 30720 il valore 33792.

Questo libro è stato scritto sia per il principiante sia per chi ha già avuto modo di fare esperienza sul VIC 20 o su altre macchine.

Il VIC 20 è un computer estremamente aperto all'utente ed è dotato di grafica eccellente con eccezionali possibilità nel campo dell'acustica per una macchina della sua fascia di prezzo. Questo libro vi mostra come potete ottenere il massimo da queste caratteristiche. Se volete semplicemente introdurre una breve routine ed osservare poi i colori che ballano sul video ed ascoltarne i suoni, questo testo vi permetterà di divertirvi molto. Se d'altro canto volete imparare velocemente a scrivere dei programmi vostri, questo libro vi mostra come fare. Presentandovi gli aspetti più spettacolari del VIC 20, questo libro vi porta ben avanti nella strada della più valida programmazione. Spiega in dettaglio e con abbondanza di esempi come usare sul VIC 20 le principali istruzioni del BASIC e vi fornisce una valida gamma di routine che costituiscono una scorciatoia affidabile ed efficace ad un livello più competente di programmazione.

L'AUTORE
Owen Bishop è uno scrittore ed un programmatore rinomato. Ha al suo attivo circa trenta libri, dei quali buona parte trattano di programmazione in termini accessibili a tutti. Collabora regolarmente a varie riviste inglesi che si occupano di computer.

ISBN 80 7081 167 0
L. 19.000 (IVA inclusa)

Guida al Commodore VIC 20

O. Bishop